

# LIS User's Guide

July 16, 2004

Revision 4.2

History:

Revision	Summary of Changes	Date
4.2	AGRMET supported version	April 11, 2007
4.1	LIS version 4.0.2 release	March 14, 2005
4.0	Milestone "K" submission	February 11, 2005
3.2	LIS version 3.1 release	December 17, 2004
3.1	Milestone "G" release	July 16, 2004
3.0	Milestone "G" submission	May 7, 2004
2.3	Improvements to Milestone "T"	November 30, 2003
2.2	-	-
2.1	Milestone "T" release	November 10, 2003
2.0	Milestone "T" submission	August 14, 2003
1.1	Milestone "F" release	April 25, 2003
1.0	Milestone "F" submission	March 31, 2003



National Aeronautics and Space Administration  
Goddard Space Flight Center  
Greenbelt, Maryland 20771

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	What's New . . . . .	4
1.1.1	LIS 4.1-4.2 . . . . .	4
1.1.2	LIS 4.0.2 – 4.1 . . . . .	5
1.1.3	LIS 4.0 – 4.0.2 . . . . .	5
1.1.4	LIS 3.1 – 4.0 . . . . .	5
1.1.5	LIS 3.0 – 3.1 . . . . .	5
1.1.6	LIS 2.0 – 3.0 . . . . .	6
<b>2</b>	<b>Background</b>	<b>7</b>
2.1	LIS . . . . .	7
2.2	LIS core . . . . .	7
2.3	GrADS-DODS Server . . . . .	8
<b>3</b>	<b>Preliminaries</b>	<b>9</b>
<b>4</b>	<b>Obtaining the Source Code</b>	<b>11</b>
4.1	Downloading the Source Code . . . . .	11
4.2	Source files . . . . .	11
4.3	Scripts . . . . .	13
<b>5</b>	<b>Obtaining The Data-sets</b>	<b>14</b>
5.1	Downloading The Data-sets . . . . .	14
<b>6</b>	<b>Building the Executable</b>	<b>15</b>
6.1	General Build Instructions . . . . .	15
6.1.1	Required Software Libraries . . . . .	16
6.1.2	Modifying the Makefile . . . . .	16
6.2	Generating documentation . . . . .	17
<b>7</b>	<b>Running The Executable</b>	<b>18</b>
7.1	Configuring Run Via LIS config File . . . . .	18
7.1.1	Overall driver options . . . . .	18
7.1.2	Choice of parameters . . . . .	20
7.1.3	Runtime options . . . . .	22
7.1.4	Processor layout . . . . .	25
7.1.5	Running domain section . . . . .	25
7.2	Domain Example . . . . .	26
7.2.1	Parameter domain . . . . .	28
7.2.2	Landcover data specification . . . . .	32
7.2.3	Topography data specification . . . . .	32
7.2.4	soil parameter data specification . . . . .	33
7.2.5	Albedo data specification . . . . .	34
7.2.6	Greenness fraction data specification . . . . .	34
7.2.7	LAI/SAI data specification . . . . .	34

7.2.8	GEOS forcing specification . . . . .	34
7.2.9	GDAS forcing specification . . . . .	35
7.2.10	NLDAS forcing specification . . . . .	35
7.2.11	ECMWF forcing specification . . . . .	35
7.2.12	BERG forcing specification . . . . .	35
7.2.13	AGRMET forcing specification . . . . .	36
7.2.14	CMAP forcing specification . . . . .	37
7.2.15	Huffman forcing specification . . . . .	38
7.2.16	CMORPH forcing specification . . . . .	38
7.2.17	Noah LSM specification . . . . .	38
7.2.18	CLM LSM specification . . . . .	39
7.2.19	Model output configuration . . . . .	40
<b>8</b>	<b>Output Data Processing</b>	<b>42</b>
<b>A</b>	<b>LIS config File</b>	<b>43</b>
<b>B</b>	<b>Makefile</b>	<b>52</b>
<b>C</b>	<b>READ_GRIB - Information and Instructions</b>	<b>63</b>
C.1	SUBROUTINE GRIBGET (NUNIT, IERR) . . . . .	63
C.2	SUBROUTINE GRIBREAD (NUNIT, DATA, NDATA, IERR) . . . . .	64
C.3	SUBROUTINE GRIBHEADER (IERR) . . . . .	64
C.4	SUBROUTINE GRIBDATA (DATARRAY, NDAT) . . . . .	64
C.5	SUBROUTINE GRIBPRINT (ISEC) . . . . .	65
C.6	SUBROUTINE GET_SEC1 (KSEC1) . . . . .	65
C.7	SUBROUTINE GET_SEC2 (KSEC2) . . . . .	65
C.8	SUBROUTINE GET_GRIDINFO (IGINFO, GINFO) . . . . .	65
C.9	C-ROUTINE COPEN (UNIT, NUNIT, NAME, MODE, ERR, OFLAG) . . . . .	65
C.10	SEC Header Array Information Tables . . . . .	66
C.11	Additional information for setting up the READ_GRIB routines for use on Linux Machines . . . . .	68
C.12	Example of Fortran code that calls READ_GRIB routines . . . . .	69

# 1 Introduction

This is the LIS' User's Guide. This document describes how to download and install the code and data needed to run the LIS executable for LIS' "CHANGE-Customer Delivery" revision 4.2. It describes how to build and run the code, and finally this document also describes how to download output data-sets to use for validation. Updates to this document will provide more detailed instructions on how to configure the executable and will address the graphical user interface.

This document consists of 8 sections, described as follows:

- 1 Introduction:** the section you are currently reading
- 2 Background:** general information about the LIS project
- 3 Preliminaries:** general information, steps, instructions, and definitions used throughout the rest of this document
- 4 Obtaining the Source Code:** the steps needed to download the source code
- 5 Obtaining the Data-sets:** the steps needed to download the data-sets
- 6 Building the Executable:** the steps needed to build the LIS executable
- 7 Running the Executable:** the steps needed to prepare and submit a run, also describes the various run-time configurations
- 8 Output Data Processing:** the steps needed to post-process generated output for visualization

## 1.1 What's New

See *RELEASE\_NOTES* found in the *source.tar.gz* file for more details. (See Section 4.)

### 1.1.1 LIS 4.1-4.2

1. Completed implementation of AGRMET processing algorithms
2. Ability to run on polar stereographic, mercator, lambert conformal, and lat/lon projections
3. Updated spatial interpolation tools to support the transformations to/from the above grid projections
4. Switched to a highly interactive configurations management from the fortran namelist-based lis.crd style.
5. Streamlined error and diagnostic logging, in both sequential and parallel processing environments.

6. extended grib support; included the UCAR-based read-grib library
7. Support for new supplemental forcing analyses - Huffman, CMORPH

#### **1.1.2 LIS 4.0.2 – 4.1**

1. Preliminary AFWA support
2. Ability to run on a defined layout of processors.
3. Updates to plugins, preliminary implementation of alarms.
4. Definition of LIS specific environment variables.

#### **1.1.3 LIS 4.0 – 4.0.2**

1. GSWP-2 support – LIS can now run GSWP-2 experiments. Currently only CLM and Noah models have full support.
2. Updates to the 1km running mode.
3. Updates to the GDS running mode.

#### **1.1.4 LIS 3.1 – 4.0**

1. VIC 4.0.5 – LIS' implementation of VIC has been reinstated.

#### **1.1.5 LIS 3.0 – 3.1**

1. New domain-plugin support – facilitates creating new domains.
2. New domain definition support – facilitates defining running domains. Sub-domain selection now works for both MPI-based and non MPI-based runs.
3. New parameter-plugin support – facilitates adding new input parameter data-sets.
4. New LIS version of ipolates – facilitates creating new domains and base forcing data-sets.
5. Compile-time MPI support – MPI libraries are no longer required to compile LIS.
6. Compile-time netCDF support – netCDF libraries are no longer required to compile LIS.
7. New LIS time manager support – ESMF time manager was removed. ESMF libraries are not required in this version of LIS.

### **1.1.6 LIS 2.0 – 3.0**

1. Running Modes – Now there is more than one way to run LIS. In addition to the standard MPI running mode, there are the GDS running mode and the 1 km running mode.
2. Sub-domain Selection – Now you are no longer limited to global simulations. You may choose any sub-set of the global domain to run over. See Section 7.1.5 and Section 7.2.1 for more details. (This is currently only available for the MPI-based running mode.)
3. Plug-ins – Now it is easy to add new LSM and forcing data-sets into the LIS driver. See LIS’ Developer’s Guide for more details.

## 2 Background

This section provides some general information about the LIS project and land surface modeling.

### 2.1 LIS

The primary goal of the LIS project is to build a system that is capable of performing high resolution land surface modeling at high performance using scalable computing technologies. The LIS software system consists of a number of components: (1) LIS driver: the core software that integrates the use of land surface models, data management techniques, and high performance computing. (2) community land surface models such as CLM [3], and Noah [5] and (3) Visualization and data management tools such as GrADS [1] -DODS [4] server. One of the important design goals of LIS is to develop an interoperable system to interface and interoperate with land surface modeling community and other earth system models. LIS is designed using an object oriented, component-based style. The adaptable interfaces in LIS can be used by the developers to ease the cost of development and foster rapid prototyping and development of applications. The following sections describe the main components of LIS.

### 2.2 LIS core

The central part of LIS software system is the LIS core that controls program execution. The LIS core is a model control and input/output system (consisting of a number of subroutines, modules written in Fortran 90 source code) that drives multiple offline one-dimensional LSMs. The one-dimensional LSMs such as CLM and Noah, apply the governing equations of the physical processes of the soil-vegetation-snowpack medium. These land surface models aim to characterize the transfer of mass, energy, and momentum between a vegetated surface and the atmosphere. When there are multiple vegetation types inside a grid box, the grid box is further divided into “tiles”, with each tile representing a specific vegetation type within the grid box, in order to simulate sub-grid scale variability.

The execution of the LIS core starts with reading in the user specifications, including the modeling domain, spatial resolution, duration of the run, etc. Section 7 describes the exhaustive list of parameters specified by the user. This is followed by the reading and computing of model parameters. The time loop begins and forcing data is read, time/space interpolation is computed and modified as necessary. Forcing data is used to specify the boundary conditions to the land surface model. The LIS core applies time/space interpolation to convert the forcing data to the appropriate resolution required by the model. The selected model is run for a vector of “tiles” and output and restart files are written at the specified output interval.

Some of the salient features provided by the LIS core include:

- Vegetation type-based “tile” or “patch” approach to simulate sub-grid scale variability.
- Makes use of various satellite and ground-based observational systems.
- Derives model parameters from existing topography, vegetation, and soil coverages.
- Extensible interfaces to facilitate incorporation of new land surface models, forcing schemes.
- Uses a modular, object oriented style design that allows “plug and play” of different features by allowing user to select only the components of interest while building the executable.
- Ability to perform regional modeling (only on the domain of interest).
- Provides a number of scalable parallel processing modes of operation.

Please refer to the software design document for a detailed description of the design of LIS core. The LIS reference manual provides a description of the extensible interfaces in LIS. The “plug and play” feature of different components is described in this document.

### **2.3 GrADS-DODS Server**

A GrADS-DODS Server (GDS) is a data server built upon the Grid Analysis and Display System (GrADS) and the Distributed Oceanographic Data System (DODS).

GrADS is an earth science data manipulation and visualization tool under development at the Center for Ocean-Land-Atmosphere Studies (COLA) (<http://grads.iges.org/cola.html>). See <http://grads.iges.org/grads/grads.html> for more detailed information about GrADS.

DODS, also called the Open source Project for a Network Data Access Protocol (OPeNDAP), is a protocol for serving data-sets stored in various formats over a network. See <http://www.unidata.ucar.edu/packages/dods/> for more detailed information about DODS.

A GDS may be used to provide the LIS core with the forcing and input parameter data needed to run an LSM.

### 3 Preliminaries

This code has been compiled and run on Linux PC (Intel/AMD based) systems, IBM AIX systems, SGI IRIX64 6.5 systems, and SGI Altix system. These instructions expect that you are using such a system. In particular you need

Software:

- Linux
  - Absoft's Pro Fortran Software Development Kit, version 8.0
    - or
    - Lahey/Fujitsu's Fortran 95 Compiler, release L6.00c
  - GNU's C and C++ compilers, gcc and g++, version 2.96
  - MPICH , version 1.2.5.2
  - GNU's make, gmake, version 3.77
- IBM
  - XL Fortran version 8 release 1
  - GNU's make, gmake, version 3.77
- SGI IRIX
  - MIPSpro version 7.3.1.1m
  - Message Passing Toolkit, mpt, version 1.5.3.0
  - GNU's make, gmake, version 3.77
- SGI Altix
  - Intel Fortran Compiler version 7
  - Message Passing Toolkit, mpt, version 1.9.1.0
  - GNU's make, gmake, version 3.77
- Compaq Alpha
  - HP Fortran V5.5
  - Message Passing Toolkit, mpt, version 1.9.1.0
  - GNU's make, gmake, version 3.77

System Resources:

	1/4 deg	5 km	1 km
memory	250 MB	32 GB	800 MB (per patch)
hard disk space	3 GB	46 GB	850 GB
source	64 MB	64 MB	64 MB
input data	1.5 GB	25 GB	200 GB
output data	1 GB	20 GB	640 GB

Note, the requirements for input and output data are for a 1-day simulation.

You need to create a working directory on your system that has sufficient disk space to install and run in. Throughout the rest of this document this directory shall be referred to as *\$WORKING*.

## 4 Obtaining the Source Code

This section describes how to obtain the source code needed to build the LIS executable.

### 4.1 Downloading the Source Code

To obtain the source code needed for LIS’ “CHANGE–Customer Delivery” revision 4.2:

1. Go to LIS’ “Public Release Home Page”  
Go to <http://lis.gsfc.nasa.gov/>  
Follow the “Source Codes” link.  
Follow the “LIS 4.2 Code Release” link.
2. From LIS’ “Public Release Home Page”  
Follow the “LIS 4.2 Source Code and Scripts” link.
3. Download the required *source.tar.gz* and *scripts.tar.gz* files into your working directory, *\$WORKING/LIS*.  
Use the “LIS driver and Land Surface Models” link to get *source.tar.gz*.  
Use the “LIS scripts” link to get *scripts.tar.gz*.
4. Unpack these files. Run (in the order listed):  

```
% gzip -dc source.tar.gz | tar xf -
% gzip -dc scripts.tar.gz | tar xf -
```

Unpacking the *scripts.tar.gz* file will also create the input directory tree needed for downloading the input data-sets.

### 4.2 Source files

Unpacking the *source.tar.gz* file will create a *\$WORKING/LIS/src* sub-directory. The structure of *src* is as follows:

Directory Name	Synopsis
baseforcing	Top level directory for base forcing methods
baseforcing/berg	Routines for handling ECMWF reanalysis forcing product
baseforcing/ecmwf	Routines for handling ECMWF forcing product
baseforcing/geos	Routines for handling GEOS forcing product
baseforcing/gdas	Routines for handling GDAS forcing product
baseforcing/nldas	Routines for handling NLDAS forcing product
baseforcing/afwa	Routines for handling AFWA forcing
core	LIS core routines
plugins	Modules defining the function table registry of extensible functionalities
domains	Top level directory for domain implementations
domains/latlon	Routines for creating grid and tile spaces in a lat/lon projection
domains/lambert	Routines for creating grid and tile spaces in a lambert conformal projection
domains/merc	Routines for creating grid and tile spaces in a mercator projection
domains/polar	Routines for creating grid and tile spaces in a polar stereographic projection
domains/gswp	Routines for creating grid and tile spaces for a GSWP simulation
domains/afwa	Routines for creating grid and tile spaces for a AFWA-AGRMET simulation
interp	Interpolation routines
lib	Libraries needed for linking
lsms/clm2	Top level clm2 land surface model sub-directory
lsms/clm2/biogeochem	Biogeochemistry routines
lsms/clm2/biogeophys	Biogeophysics routines (e.g., surface fluxes)
lsms/clm2/camclm_share	Code shared between the clm2 and cam (e.g., calendar information)
lsms/clm2/csm_share	Code shared by all the geophysical model components of the Community Climate System Model (CCSM). Currently contains code for CCSM message passing orbital calculations and system utilities
lsms/clm2/ecosysdyn	Ecosystem dynamics routines (e.g., leaf and stem area index)
lsms/clm2/main	Control (driver) routines
lsms/clm2/mksrfdata	Routines for generating surface data-sets
lsms/clm2/utils	Independent utility routines
lsms/hyssib	hyssib land surface model
lsms/mosaic	mosaic land surface model
lsms/noah.2.6	Noah land surface model version 2.6
lsms/noah.2.7	Noah land surface model version 2.7.1
lsms/ssib	ssib land surface model
lsms/vic	VIC land surface model

make	Makefile and needed headers
suppforcing	Top level directory for supplemental products
suppforcing/cmap	Routines for handling CMAP precipitation product
suppforcing/huff	Routines for handling HUFFMAN precipitation product
suppforcing/cmor	Routines for handling CMORPH precipitation product
params	Top level directory for input parameter products
params/topo	Routines for handling topography data
params/lai	Routines for handling leaf/stem area index data
params/landcover	Routines for handling land cover classification data
params/soils	Routines for handling soil classification data
params/tbot	Routines for handling bottom temperature data
params/albedo	Routines for handling albedo data
params/gfrac	Routines for handling greenness fraction data
params/snow	Routines for handling snow data
runmodes	Top level directory for different running modes
runmodes/retrospective	Routines to manage a retrospective mode
runmodes/agrmemode	Routines to manage an AFWA/AGRMET mode
tables	Contains the GRIB tables for writing grib output

Source code documentation may be found on LIS' web-site on LIS' "Public Release Home Page". Follow the "LIS 4.2 Source Code Documentation" link.

### 4.3 Scripts

The *scripts.tar.gz* file contains a script for compiling and building the executable and a sample card file used for running the LIS executable and for configuring the individual runs. These are described in Section 7.

Unpacking the *scripts.tar.gz* file will place the following files into the *\$WORKING/LIS* sub-directory:

File Name	Synopsis
lis.config	Sample configuration file
comp.csh	Compile and build script
utils	The in-line documentation processing scripts
input	An empty directory tree to hold the input data

## **5 Obtaining The Data-sets**

This section describes how to obtain sample data-sets for testing the LIS executable.

### **5.1 Downloading The Data-sets**

To obtain sample data-sets for LIS’ “CHANGE–Customer Delivery” revision 4.2:

1. Go to LIS’ “Home Page”  
Go to <http://lis.gsfc.nasa.gov/>
2. Follow the “Get LIS Data” link.  
For specific data needs, please contact the LIS team.

## 6 Building the Executable

This section describes how to build the source code and create LIS' executable – named LIS.

1. Perform the steps described in Section 4.
2. Define/set the LIS-specific environmental variables based on the platform configuration. LIS\_ARCH - for the architecture
  - linux\_absoft : Linux with absoft compiler
  - linux\_pgi : Linux with portland compiler
  - linux\_lf95 : Linux with lahey compiler
  - linux\_ifc : Linux with intel compiler
  - AIX : IBM-SP
  - OSF1 : Comaq alpha
  - IRIX64 : SGI
- LIS\_SPMD - defines the computing mode
  - single : single processor, no MPI
  - parallel : compile with MPI support
3. Compile the new GRIB library, *libw3.a*. You must edit the *Makefile* located in *\$WORKING/LIS/src/lib/w3lib*. Uncomment the appropriate block of compiler flags, then run **gmake**.
4. Compile the new GRIB library, *read\_grib*. You must edit the *Makefile* located in *\$WORKING/LIS/src/lib/read\_grib*. Uncomment the appropriate block of compiler flags, then run **gmake**. Please refer to the Appendix C for helpful suggestions and instructions.
5. Compile the LIS source code.
  - (a) Change directory into *\$WORKING/LIS/*.  
% cd \$WORKING/LIS/
  - (b) Run the compiling script.  
% ./comp.csh

See Appendix B to see the Makefile.

### 6.1 General Build Instructions

This section describes how to build the LIS code on a platform other than those discussed in Section 3.

### **6.1.1 Required Software Libraries**

In order to build the LIS executable, the following libraries must be installed on your system:

- Message Passing Interface (MPI) – If you wish to run the MPI-based running mode
  - vendor supplied, or
  - MPICH
    - (<http://www-unix.mcs.anl.gov/mpi/mpich/>)
- w3lib – Source include in the *source.tar.gz* file.

To install the MPI libraries, follow the instructions provided at the MPI URL listed above.

Note: Due to the mix of programming languages (Fortran and C) used by LIS, you may run into linking errors when building the LIS executable.

When compiling code using Absoft's Pro Fortran SDK, set the following compiler options:

`-YEXT_NAMES=LCS -s -B108 -YCFRL=1`

These must be set for each of the above libraries.

### **6.1.2 Modifying the Makefile**

This section lists the variables in the *Makefile* that must be set by the user before compiling.

Variable	Description
<b>UNAMES</b>	set by call to <code>uname</code> to determine what type of system you are using. Determine which variable ( <b>UNAMES</b> or <b>UMACHINE</b> ) will contain the needed information and use it.
<b>UMACHINE</b>	set by call to <code>uname</code> to determine what type of system you are using. Determine which variable ( <b>UNAMES</b> or <b>UMACHINE</b> ) will contain the needed information and use it.
<b>MPI_PREFIX</b>	path to where mpi libraries are installed
<b>LIB_MPI</b>	path to mpi libraries
<b>INC_MPI</b>	path to mpi header files
<b>ESMF_DIR</b>	path to where esmf libraries are installed
<b>LIB_ESMF</b>	path to esmf libraries
<b>MOD_ESMF</b>	path to esmf modules
<b>ESMF_ARCH</b>	system on which esmf libraries were compiled
<b>FC</b>	fortran compiler
<b>CPP</b>	C preprocessor
<b>LIB_DIR</b>	path to where lis libraries are installed
<b>CPPFLAGS</b>	flags for C preprocessor
<b>CFLAGS</b>	flags for C compiler
<b>FFLAGS</b>	flags for Fortran compiler
<b>FOPTS</b>	additional options for compiler and linker
<b>LDFLAGS</b>	flags for linker
<b>NEW_ARCH_HERE</b>	replace this with the type of system you are using, either the result from <code>uname -s</code> or <code>uname -m</code> . E.g., IRIX64 or i686

Note: For Linux architectures, the default **ESMF\_ARCH** is set to be `linux_absoft`.  
For Linux systems using the Lahey Fortran compiler, **ESMF\_ARCH** must be changed to `linux_lf95`.

## 6.2 Generating documentation

LIS code uses the ProTex documenting system [2]. The documentation in **LATEX** format can be produced by typing `gmake doc` in the `$WORKING/LIS/src/make` directory. This command produces documentation, generating all the files in `$WORKING/LIS/doc` directory. These files can be easily converted to pdf or html formats using utilites such as `pdflatex` or `latex2html`.

## 7 Running The Executable

This section describes how to run the LIS executable. Once the LIS executable is built, a simulation can be performed using the *lis.config* file.

The single-process version of LIS is executed by the following command issued in the *\$WORKING/LIS* directory.

```
% ./LIS
```

The parallel version of LIS must be run through an *mpirun* script. Assuming that MPI is installed correctly, the LIS simulation is carried out by the following command issued from in the *\$WORKING/LIS* directory.

```
% mpirun -np N ./LIS
```

The *-np N* flag indicates the number of processes to use in the run, where you replace *N* with the number of processes to use. On a multiprocessor machine, the parallel processing capabilities of LIS can be exploited using this flag.

### 7.1 Configuring Run Via LIS config File

This section describes how to configure your LIS run by specifying the options in the *lis.config* file.

See Appendix A to see a sample lis config file.

#### 7.1.1 Overall driver options

The overall driver options consists of the following:

**Running mode:**

**Domain type:**

**Number of subnests:**

**Land surface model:**

**Base forcing source:**

**Supplemental forcing source:**

**Running mode:** specifies the running mode used in LIS. Acceptable values are:

Value	Description
1	Retrospective running mode
2	AGMRET operational running mode

**Domain type:** specifies the LIS domain used for the run Acceptable values are:

Value	Description
1	Lat/Lon projection with SW to NE data ordering
2	GSPW domain
3	Polar stereographic projection with SW to NE data ordering
4	Lambert conformal projection with SW to NE data ordering
5	Mercator projection with SW to NE data ordering
6	AFWA lat/lon 0.5 degree domain with no subgrid tiling (used for the benchmarks)

**Number of subnests:** specifies the number of subnests used for the run. Values 0 or higher area acceptable. The maximum number of subnests is limited by the amount of available memory on the system. The specifications for different nests are done using white spaces as the delimiter. Please see below for further explanations. Note that all nested domains should run on the same projection and same land surface model.

**Land surface model:** specifies the land surface model to run. Acceptable values are:

Value	Description
1	Noah
2	CLM
3	VIC
4	mosaic
5	hyssib
6	ssib
7	catchment

**Base forcing source:** specifies the forcing data source for the run. Acceptable values are:

Value	Description
1	GDAS
2	GEOS
3	ECMWF
4	NLDAS
5	GSPW
6	BERG
7	AGRMET

**Supplemental forcing source:** specifies the supplemental forcing data source for the run. Acceptable values are:

Value	Description
0	None
2	CMAP
11	Huffman
14	CMORPH

### 7.1.2 Choice of parameters

The parameter choice specification consists of the following:

Landcover data source:  
Use soil texture:  
Soil data source:  
Soil color data source:  
Elevation data source:  
LAI data source:  
Albedo data source:  
Greenness data source:  
Porosity data source:  
Ksat data source:  
B parameter data source:  
Quartz data source:

**Landcover data source:** specifies the usage of soils data in the run. Acceptable values are:

Value	Description
1	use the UMD landcover
2	use the USGS landcover data

**Use soil texture:** specifies the usage of soil texture data in the run. Acceptable values are:

Value	Description
1	use a texture map
0	use a sand/silt/clay percentage map

**Soil data source:** specifies the source of soil parameters in the run. Acceptable values are:

Value	Description
1	use FAO based maps
2	use STATSGO based maps

**Soil color data source:** specifies the source of soil color data in the run. Acceptable values are:

Value	Description
0	do not use soil color
1	use FAO based map

**Topography data source:** specifies topography data source for the run. Acceptable values are:

Value	Description
0	do not use
1	gtopo30 based

**LAI data source:** specifies the LAI data source for the run. Acceptable values are:

Value	Description
0	do not use
2	AVHRR-based LAI
3	MODIS-based LAI

**Albedo data source:** specifies if albedo data is to be used in the run. Acceptable values are:

Value	Description
0	Do not read albedo data
1	Read albedo data

**Greenness data source:** specifies if greenness fraction data is to be used in the run. Acceptable values are:

Value	Description
0	Do not read gfrac data
1	Read gfrac data

**Porosity data source:** specifies if soil porosity data is to be used in the run. Acceptable values are:

Value	Description
0	Do not read porosity data
1	Read porosity data

**Ksat data source:** specifies if hydraulic conductivity data is to be used in the run. Acceptable values are:

Value	Description
0	Do not read ksat data
1	Read ksat data

**B parameter data source:** specifies if the b parameter data is to be used in the run. Acceptable values are:

Value	Description
0	Do not read b parameter data
1	Read b parameter data

**Quartz data source:** specifies if the quartz data is to be used in the run. Acceptable values are:

Value	Description
0	Do not read quartz data
1	Read quartz data

### 7.1.3 Runtime options

In the `Runtime options` segment of the configuration file these parameters may be reset:

**Experiment code:**  
**Number of veg types:**  
**Number of forcing variables:**  
**Use elevation correction:**  
**Spatial interpolation method:**  
**Temporal interpolation method:**  
**Output forcing:**  
**Output methodology:**  
**Output data format:**  
**Logging level:**  
**Start mode:**  
**Starting year:**  
**Starting month:**  
**Starting day:**  
**Starting hour:**  
**Starting minute:**  
**Starting second:**  
**Ending year:**  
**Ending month:**  
**Ending day:**  
**Ending hour:**  
**Ending minute:**  
**Ending second:**  
**Model timestep:**  
**Undefined value:**  
**Output directory:**  
**Diagnostic output file:**  
**Number of ensembles per grid:**

**Experiment code:** specifies the “experiment code number” for the run. It is used in constructing the name of the output directory for the run. Acceptable values are any name using upto 3 characters.

**Number of veg types:** specifies the number of vegetation types used in the landcover data.

**Number of forcing variables:** specifies the number of forcing variables. LIS currently uses 10 variables to force the LSMs

**Use elevation correction:** specifies whether to use elevation correction for forcing.

Acceptable values are:

Value	Description
0	Do not use elevation correction for forcing
1	Use elevation correction for forcing

**Spatial interpolation method:** specifies the type of interpolation scheme to apply to the forcing data. Acceptable values are:

Value	Description
1	bilinear scheme
2	conservative scheme

**Temporal interpolation method:** specifies the type of temporal interpolation scheme to apply to the forcing data. Acceptable values are:

Value	Description
1	next scheme
2	ueber next scheme

**Output forcing:** specifies whether to output the ALMA optional forcing variables. Acceptable values are:

Value	Description
0	Do not output forcing variables
1	Do output forcing variables

**Output methodology:** specifies whether to write output as a 1-D array containing only land points or as a 2-D array containing both land and water points. Acceptable values are:

Value	Description
0	Do not write output
0	Write output in a 1-D grid domain
1	Write output in a 2-D grid domain

**Output data format:** specifies the format of the model output data. Acceptable values are:

Value	Description
1	Write output in binary format
2	Write output in Grib format
3	Write output in NETCDF format

**Logging level:** specifies different levels of logging. Acceptable values are:

Value	Description
1	Default messages
2	Default messages and debugging messages

**Start mode:** specifies if a restart mode is being used. Acceptable values are:

Value	Description
1	A restart mode is being used
2	A cold start mode is being used, no restart file read

When the cold start option is specified, the program is initialized using the LSM-specific initial conditions (typically assumed uniform for all tiles). When a restart mode is used, it is assumed that a corresponding restart file is provided depending upon which LSM is used. The user also needs to make sure that the ending time of the simulation is greater than model time when the restart file was written.

The start time is specified in the following format:

Variable	Value	Description
Starting year:	integer 2001 – present	specifying starting year
Starting month:	integer 1 – 12	specifying starting month
Starting day:	integer 1 – 31	specifying starting day
Starting hour:	integer 0 – 23	specifying starting hour
Starting minute:	integer 0 – 59	specifying starting minute
Starting second:	integer 0 – 59	specifying starting second

The end time is specified in the following format:

Variable	Value	Description
Ending year:	integer 2001 – present	specifying ending year
Ending month:	integer 1 – 12	specifying ending month
Ending day:	integer 1 – 31	specifying ending day
Ending hour:	integer 0 – 23	specifying ending hour
Ending minute:	integer 0 – 59	specifying ending minute
Ending second:	integer 0 – 59	specifying ending second

**Model timestep:** specifies the time-step for the run. Acceptable values are:

Value	Description
900	15 minute time-step
1800	30 minute time-step
3600	60 minute time-step

For a nested domain, the timesteps for each nest should be specified with white spaces as the delimiter. If two domains (one subnest) are employed, the first one using 900 seconds and the second one using 3600 seconds as the timestep, the model timesteps are specified as:

`Model timestep: 900 3600`

**Undefined value:** specifies the undefined value. The default is set to -9999.

**Output directory:** specifies the name of the top-level output directory. Acceptable values are any 40 character string. The default value of is set to OUTPUT. For simplicity, throughout the rest of this document, this top-level output directory shall be referred to by its default name, *\$WORKING/LIS/OUTPUT*.

**Diagnostic output file:** specifies the name of run time diagnostic file. Acceptable values are any 40 character string.

**Number of ensembles per grid:** specifies the number of ensembles of tiles to be used. The value should be greater than equal to 1.

The following options are set for defining the subgrid scale tiling.

**Maximum number of tiles per grid:** defines the maximum tiles per grid (this can be as many as the total number of vegetation types. In addition, users select the smallest percentage of a cell for which to create a tile. **Cutoff percentage:** defines this parameter. The percentage value is expressed as a fraction.

#### 7.1.4 Processor layout

This section specifies the 2-d layout of the processors in a parallel processing environment. The user can specify the number of processors along the east-west dimension and north-south dimension using **Number of processors along x:** and **Number of processors along y:**, respectively. Note that the layout of processors should match the total number of processors used. For example, if 8 processors are used, the layout can be specified as 1x8, 2x4, 4x2, or 8x1.

#### 7.1.5 Running domain section

This section of the config file specifies the running domain (domain over which the simulation is carried out). For a lat/lon domain, the values are defined as follows:

Variable	Description
<code>run domain lower left lat:</code>	Latitude of the south-west grid-cell center for the running domain
<code>run domain lower left lon:</code>	Longitude of the south-west grid-cell center for the running domain
<code>run domain upper right lat:</code>	Latitude of the north-east grid-cell center for the running domain
<code>run domain upper right lon:</code>	Longitude of the north-east grid-cell center for the running domain
<code>run domain resolution dx:</code>	Latitudinal increment of the running domain
<code>run domain resolution dy:</code>	Longitudinal increment of the running domain

The domain specifications for a nested run can be done similar to the example below: (A configuration for a single nest is shown below. The first domain is at 1km and the second one at 3km.)

```
run domain lower left lat:          32.995   29.985
run domain lower left lon:          -97.005  -100.005
run domain upper right lat:         33.995   39.975
run domain upper right lon:         -96.005  -90.015
run domain resolution dx:           0.01     0.03
run domain resolution dy:           0.01     0.03
```

The specification of the running domain section depends on the type of LIS domain and projection used. Some examples of other domains are shown below:

#### Polar Stereographic

```
un domain lower left lat:          34.35
run domain lower left lon:          -100.18
run domain true lat:               36.61
run domain standard lon:           -97.49
run domain orientation:            0.0
run domain resolution:             1.0 #in KMs
run domain x-dimension size:       200
run domain y-dimension size:       200
```

#### Lambert Conformal

```
run domain lower left lat:          34.35
run domain lower left lon:          -100.18
run domain true lat1:              36.61
run domain true lat2:              36.61
run domain standard lon:           -97.49
run domain resolution:              1.0 #in KMs
run domain x-dimension size:       500
run domain y-dimension size:       500
```

#### Mercator

```
run domain lower left lat:          34.35
run domain lower left lon:          -100.18
run domain true lat1:              36.61
run domain standard lon:           -97.49
run domain resolution:              1.0 #in KMs
run domain x-dimension size:       500
run domain y-dimension size:       500
```

## 7.2 Domain Example

This section describes how to compute the values for the run domain and param domain sections.

First, we shall generate the values for the parameter data domain. LIS' parameter data is defined on a Latitude/Longitude grid, from  $-180$  to  $180$  degrees longitude and from  $-60$  to  $90$  degrees latitude.

For this example, consider running at  $1/4$  deg resolution. The coordinates of the south-west and the north-east points are specified at the grid-cells' centers. Here the south-west grid-cell is given by the box  $(-180, -60), (-179.750, -59.750)$ . The center of this box is  $(-179.875, -59.875)$ .<sup>1</sup>

```
param domain lower left lat: -59.875
param domain lower left lon: -179.875
```

The north-east grid-cell is given by the box  $(179.750, 89.750), (180, 90)$ . Its center is  $(179.875, 89.875)$ .

```
param domain upper right lat: 89.875
param domain upper right lon: 179.875
```

Setting the resolution ( $0.25$  deg) gives

```
param domain resolution dx: 0.25
param domain resolution dy: 0.25
```

And this completely defines the parameter data domain.

Next, we shall generate the values for the running domain.

If you wish to run over the whole domain defined by the parameter data domain then you simply set the values defined in the parameter domain section in the run domain section. This gives:

```
run domain lower left lat: -59.875
run domain lower left lon: -179.875
run domain upper right lat: 89.875
run domain upper right lon: 179.875
run domain resolution dx: 0.25
run domain resolution dy: 0.25
```

Now say you wish to run only over the region given by  $(-97.6, 27.9), (-92.9, 31.9)$ . Since the running domain is a sub-set of the parameter domain, it is also a Latitude/Longitude domain at  $1/4$  deg. resolution. Thus,

```
run domain resolution dx: 0.25
run domain resolution dy: 0.25
```

Now, since the running domain must fit onto the parameter domain, the desired running region must be expanded from  $(-97.6, 27.9), (-92.9, 31.9)$  to  $(-97.75, 27.75), (-92.75, 32.0)$ . The south-west grid-cell for the running domain is the box  $(-97.75, 27.75), (-97.5, 28.0)$ . Its center is  $(-97.625, 27.875)$ ; giving

```
run domain lower left lat: 27.875
run domain lower left lon: -97.625
```

---

<sup>1</sup>Note, these coordinates are ordered (longitude, latitude).

The north-east grid-cell for the running domain is the box  $(-93, 31.75), (-92.75, 32.0)$ . Its center is  $(-92.875, 31.875)$ ; giving

```
run domain upper right lat: 31.875  
run domain upper right lon: -92.875
```

This completely defines the running domain.

Note, the LIS project has defined 5 km resolution to be 0.05 deg. and 1 km resolution to be 0.01 deg. If you wish to run at 5 km or 1 km resolution, redo the above example to compute the appropriate grid-cell values.

See Figure 1 for an illustration of adjusting the running grid. See Figures 2 and 3 for an illustration of the south-west and north-east grid-cells.

### 7.2.1 Parameter domain

This section describes the domain of the input parameter data grid.

Note: Several types of parameter data now have their own grid definitions. These are described in following sections.

For a lat/lon domain, the description of the parameter domain are defined as follows:

Variable	Description
param domain lower left lat:	Latitude of the south-west grid-cell center for the parameter data domain
param domain lower left lon:	Longitude of the south-west grid-cell center for the parameter data domain
param domain upper right lat:	Latitude of the north-east grid-cell center for the parameter data domain
param domain upper right lon:	Longitude of the north-east grid-cell center for the parameter data domain
param domain resolution dx:	Latitudinal increment of the parameter data domain
param domain resolution dy:	Longitudinal increment of the parameter data domain

For a nested run, the parameter domain specification should correspond to the run domain specification. Similar to the example above, the following segment shows the parameter domain specification for a single nest (1km, 3km):

```
param domain lower left lat:          -59.995  -59.985  
param domain lower left lon:          -179.995 -179.985  
param domain upper right lat:         89.995   89.985  
param domain upper right lon:         179.995  179.985  
param domain resolution dx:          0.01     0.03  
param domain resolution dy:          0.01     0.03
```

### 7.2.2 Landcover data specification

The segment of the config file specifies the land/sea mask and vegetation classification data. See Section 5 for instructions on how to obtain these parameter data-sets.

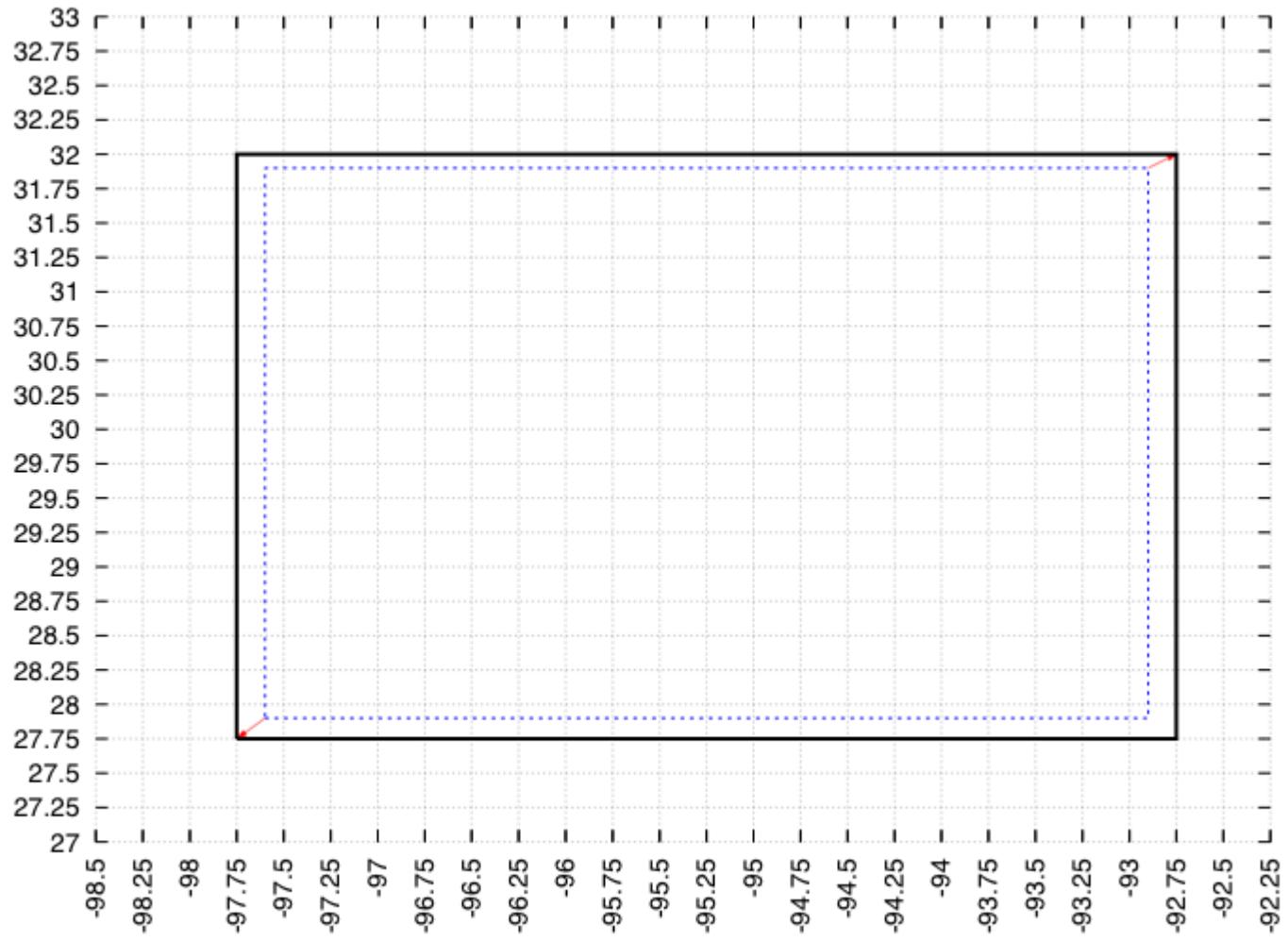


Figure 1: Illustration showing how to fit the desired running grid onto the actual grid

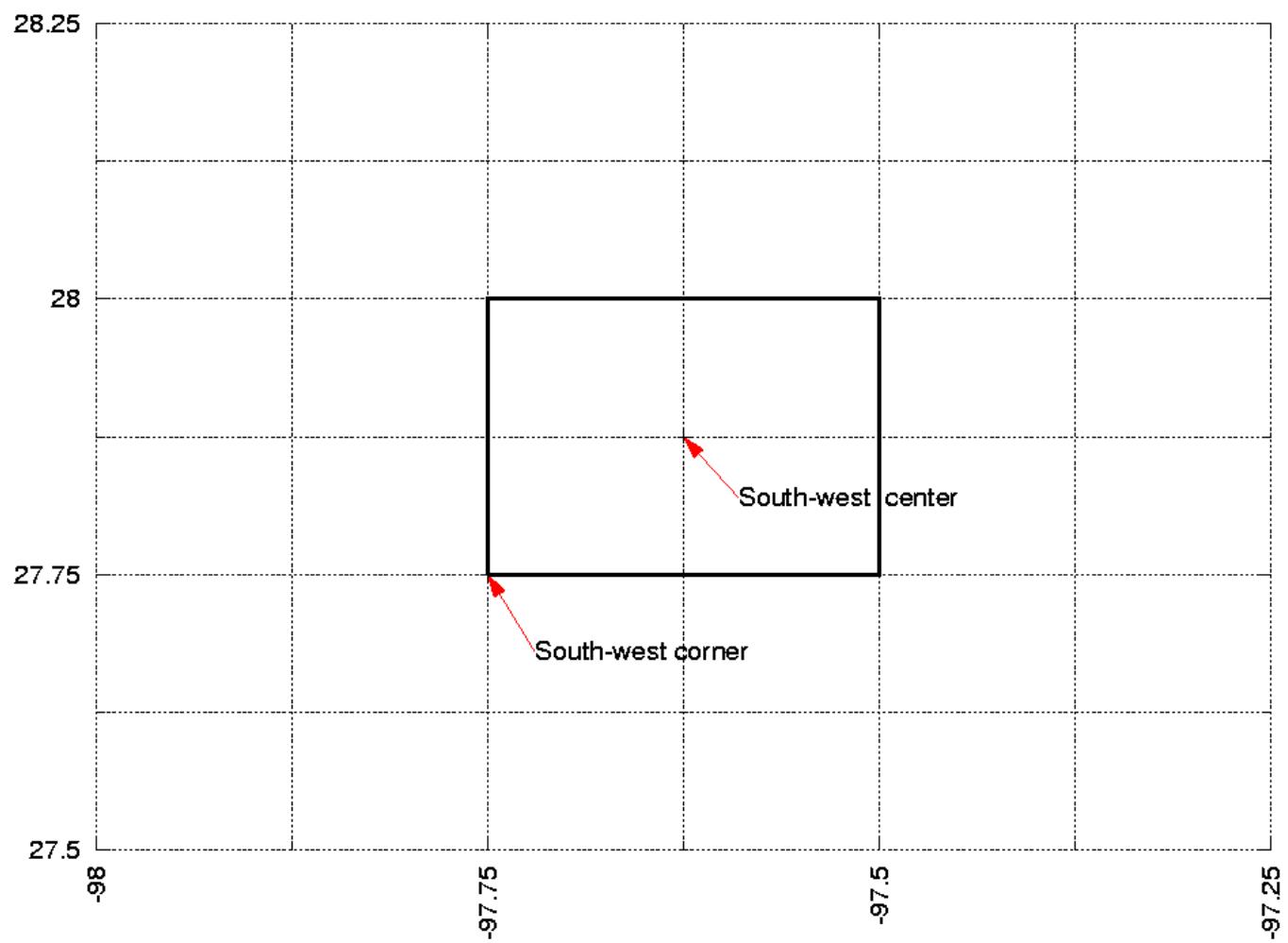


Figure 2: Illustration showing the south-west grid-cell corresponding to the example in Section 7.2

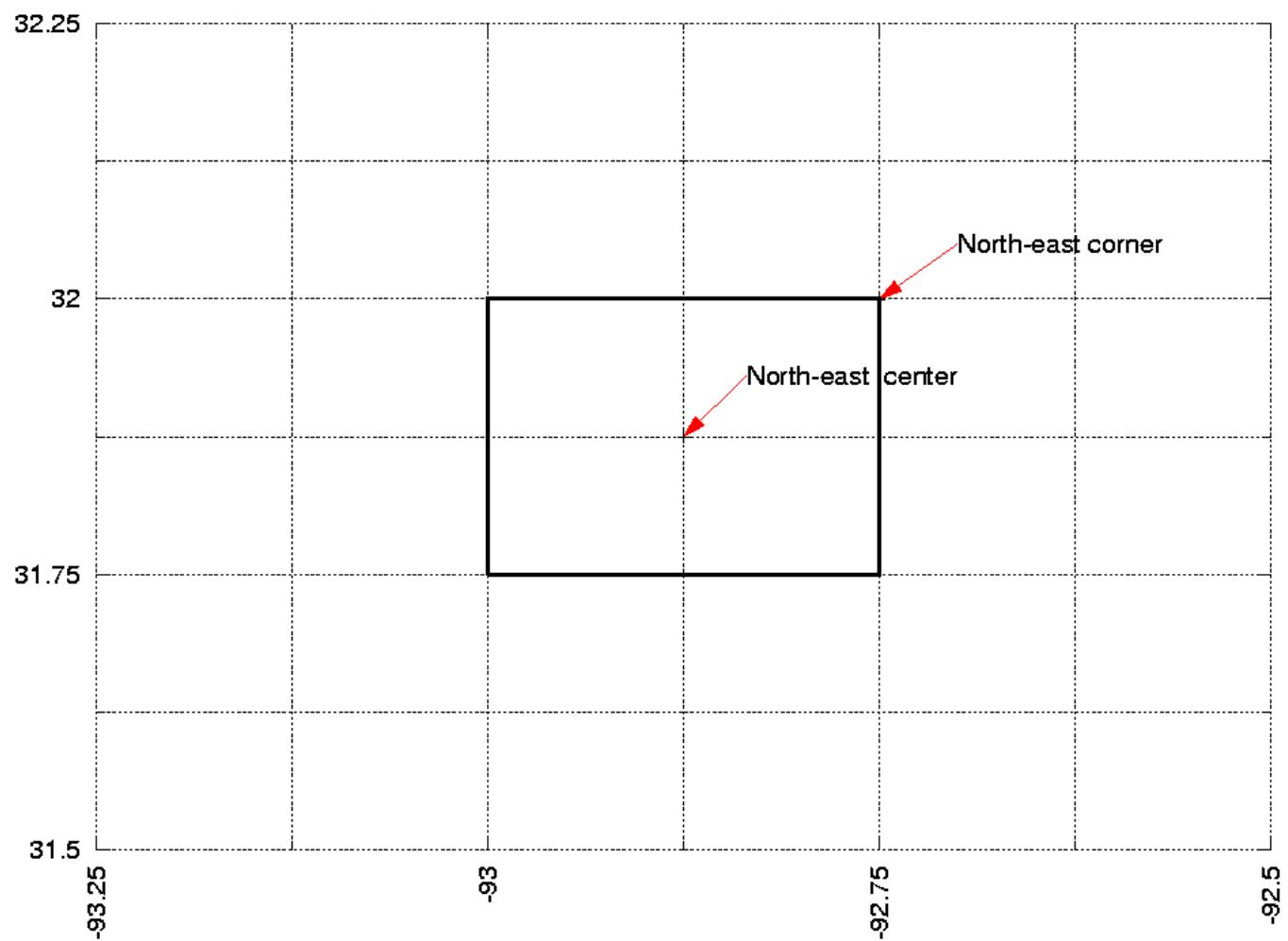


Figure 3: Illustration showing the north-east grid-cell corresponding to the example in Section 7.2

```

landmask file:
landcover file:
landcover lower left lat:
landcover lower left lon:
landcover upper right lat:
landcover upper right lon:
landcover resolution (dx):
landcover resolution (dy):

```

**landmask file:** specifies the location of land/water mask file.

**landcover file:** specifies the location of the vegetation classification file.

The remaining options define the grid these data are on. Each element is defined similarly to its corresponding element in the parameter domain definition section. See Section 7.2.1.

A nested run using different configurations of landcover/landmask can be specified similar to the following example. All other datasets (topography, soils, albedo, LAI, greenness, snow and forcings) should also specify the options for each nest in a similar manner.

```

landmask file:      ./UMD-1KM/UMD_601KMmask.1gd4r ./UMD-3KM/UMD_MASK_3KM.1gd4r
landcover file:     ./UMD-1KM/UMD_601KM.1gd4r ./UMD-3KM/UMD_VEG_3KM.1gd4r
landcover lower left lat: -59.995 -59.985
landcover lower left lon: -179.995 -179.985
landcover upper right lat: 89.995 89.985
landcover upper right lon: 179.995 179.985
landcover resolution (dx): 0.01 0.03
landcover resolution (dy): 0.01 0.03

```

### 7.2.3 Topography data specification

This segment of the config file specifies the topography data sets. See Section 5 for instructions on how to obtain these parameter data-sets.

```

elevation map:
slope map:
aspect map:
curvature map:
topography lower left lat:
topography lower left lon:
topography upper right lat:
topography upper right lon:
topography resolution (dx):
topography resolution (dy):

```

**elevation map:** specifies the elevation of the LIS grid.

**slope map:** specifies the slope of the LIS grid.  
**aspect map:** specifies the aspect of the LIS grid.  
**curvature map:** specifies the curvature of the LIS grid.

The remaining options define the grid these data are on. Each element is defined similarly to its corresponding element in the parameter domain definition. See Section 7.2.1.

#### 7.2.4 soil parameter data specification

This section of the config file specifies the soil classification data. See Section 5 for instructions on how to obtain these parameter data-sets.

```
soil texture map:  
sand fraction map:  
clay fraction map:  
silt fraction map:  
soil color map:  
porosity layer1 map:  
porosity layer2 map:  
porosity layer3 map:  
saturated matric potential map:  
saturated hydraulic conductivity map:  
b parameter map:  
quartz map:  
soils lower left lat:  
soils lower left lon:  
soils upper right lat:  
soils upper right lon:  
soils resolution (dx):  
soils resolution (dy):  
  
soil texture map: specifies the soil texture file.  
sand fraction map: specifies the sand fraction map file.  
clay fraction map: specifies the clay fraction map file.  
silt fraction map: specifies the silt map file.  
soil color map: specifies the soil color map file.  
porosity layer1 map: specifies top-layer porosity.  
porosity layer2 map: specifies mid-layer porosity.  
porosity layer3 map: specifies bottom-layer porosity.  
saturated matric potential map: specifies saturated matric potential  
saturated hydraulic conductivity map: specifies saturated hydraulic conductivity  
b parameter map: specifies b parameter  
quartz map: specifies quartz data
```

The remaining options define the grid these data are on. Each element is defined similarly to its corresponding element in the parameter domain definition section. See Section 7.2.1.

### **7.2.5 Albedo data specification**

This segment of the config file specifies the albedo data sources

**albedo map:** specifies the source of the climatology based albedo files.

**albedo climatology interval:** specifies the frequency of the albedo climatology in months. **max snow free albedo map:** specifies the static upper bound of snow free albedo file.

**bottom temperature map:** specifies the bottom boundary temperature data

### **7.2.6 Greenness fraction data specification**

This segment of the config file specifies the greenness fraction data sources

**greenness fraction map:** specifies the source of the climatology based gfrac files.

**greenness climatology interval:** specifies the frequency of the greenness climatology.

### **7.2.7 LAI/SAI data specification**

This segment of the config file specifies the Leaf/Stem Area Index data. See Section 5 for instructions on how to obtain these parameter data-sets.

**LAI map:** specifies the source of the LAI climatology files

**Snow depth map:** specifies the location of the realtime snow depth files.

### **7.2.8 GEOS forcing specification**

This segment of the config file specifies the GEOS base forcing data. See Section 5 for instructions on how to obtain these forcing data-sets.

**GEOS forcing directory:** specifies the location of the GEOS forcing files.

**GEOS domain x-dimension size:** and **GEOS domain y-dimension size:** specify the native domain parameters of the GEOS forcing data. The map projection is specified in the driver modules defined for the GEOS routines.

**GEOS number of forcing variables:** specifies the number of forcing variables provided by GEOS at the model initialization step.

### **7.2.9 GDAS forcing specification**

This segment of the config file specifies the GDAS base forcing data. See Section 5 for instructions on how to obtain these forcing data-sets.

**GDAS forcing directory:** specifies the location of the GDAS forcing files.

**GDAS elevation map:** specifies the GDAS elevation definition

**GDAS domain x-dimension size:** and **GDAS domain y-dimension size:** specify the native domain parameters of the GDAS forcing data. The map projection is specified in the driver modules defined for the GDAS routines.

**GDAS number of forcing variables:** specifies the number of forcing variables provided by GDAS at the model initialization step.

### 7.2.10 NLDAS forcing specification

This segment of the config file specifies the NLDAS base forcing data.

**NLDAS forcing directory:** specifies the location of the NLDAS forcing files.

**NLDAS elevation map:** specifies the NLDAS elevation definition

**NLDAS domain x-dimension size:** and **NLDAS domain y-dimension size:** specify the native domain parameters of the NLDAS forcing data. The map projection is specified in the driver modules defined for the NLDAS routines.

### 7.2.11 ECMWF forcing specification

This segment of the config file specifies the ECMWF base forcing data.

**ECMWF forcing directory:** specifies the location of the ECMWF forcing files.

**ECMWF domain x-dimension size:** and **ECMWF domain y-dimension size:** specify the native domain parameters of the ECMWF forcing data. The map projection is specified in the driver modules defined for the ECMWF routines.

**ECMWF number of forcing variables:** specifies the number of forcing variables provided by ECMWF at the model initialization step.

### 7.2.12 BERG forcing specification

This segment of the config file specifies the ECMWF reanalysis base forcing data.

**BERG maskfile:** specifies the location of the ECMWF reanalysis land/sea mask file.

**BERG forcing directory:** specifies the location of the BERG forcing files.

**BERG domain x-dimension size:** and **BERG domain y-dimension size:** specify the native domain parameters of the BERG forcing data. The map projection is specified in the driver modules defined for the BERG routines.

### 7.2.13 AGRMET forcing specification

This segment of the config file specifies the AGRMET base forcing data.

**AGRMET forcing directory:** specifies the location of the root directory containing the input files. The AGRMET processing algorithms assumes the following hierarchy under the root directory at each instance. For example, if the root directory for storing the files is 'FORCING/AFWA/' and the current instance is decemeber 1st, 2005, the files are stored under 'FORCING/AFWA/20051201/' directory. Under this directory the files should be stored under the following directories.

CDMS : surface and precip obs (`sfobs_*` and `preobs_*`)

GEO : GEOPRECIP files (`prec08*` and `rank08*`)

GFS : GFS data (`MT.avn*`)

PRECIP : processed precip obs (`presav_*`)

SSMI : SSM/I data (`ssmira_*`)

**SFCALC** : surface fields (sfc\*)  
**WWMCA** : WWMCA data (WWMCA\*)

**AGRMET analysis directory:** specifies the location where temporary precip analysis fields will be written

**AGRMET polar mask file:** specifies the landmask in 8th mesh polar stereographic projection used by the AGRMET algorithms.

**AGRMET polar terrain file:** specifies the terrain in 8th mesh polar stereographic projection used by the AGRMET algorithms.

**AGRMET sfcalc cntm file:** specifies the name of the files with the spreading radii used for the barnes analysis on the GFS and surface obs.

**AGRMET precip climatology:** specifies the path to the precip climatology data

**AGRMET nogaps wind weight:** specifies the weighting factor for the first guess winds

**AGRMET minimum wind speed:** specifies the minimum allowable wind speed on the AGRMET grid

**AGRMET use present/past weather estimate:** specifies whether to use present/past weather estimates (0-do not use, 1-use)

**AGRMET use precip observations:** specifies whether to use precip observations (0-do not use, 1-use)

**AGRMET use SSM/I data:** specifies whether to use SSM/I data (0-do not use, 1-use)

**AGRMET use CDFSII-based estimate:** specifies whether to use a CDFS-II based estimate (0-do not use, 1-use)

**AGRMET use GEOPRECIP estimate:** specifies whether to use a GEOPRECIP based estimate.

Acceptable values are:

Value	Description
0	do not use
1	use the estimate, do not use the rank
2	use the estimate and use the rank

**AGRMET CDFSII time interval:** specifies the CDFS-II time interval to look for cloud amount. Current value is 6.

**AGRMET use precip climatology:** specifies whether to use precip climatology. (0-do not use, 1-use)

**AGRMET SSM/I zero use switch:** specifies whether to use SSM/I zeros. (0-do not use zeros, 1-use zeros)

**AGRMET snow distribution shape parameter:** specifies the snow distribution shape parameter (A typical value is 2.6)

**AGRMET alternate monthly weighting factor:** specifies the alternate monthly weighting factor used in the precip processing.

**AGRMET minimum 3hr climo value:** specifies a minimum 3 hour precip climo value required to generate a non-zero CDFSII total cloud-based precip estimate. A typical value is 0.025.

**AGRMET maximum 3hr climo value:** specifies a maximum 3 hour precip climo value required to generate a non-zero CDFSII total cloud-based precip estimate. A typical value is 0.375.

**AGRMET minimum precip-per-precip day multiplier:** specifies a minimum precip-per-precip day multiplier used to generate a non-zero CDFSII total cloud based precip estimate. A typical value is 0.0.

**AGRMET maximum precip-per-precip day multiplier:** specifies a maximum precip-per-precip day multiplier used to generate a non-zero CDFSII total cloud based precip estimate. A typical value is 1.1

**AGRMET cloud threshold to generate CDFSII estimate:** specifies the cloud threshold to generate a CDFSII-based estimate. A typical value is 85.0.

**AGRMET median cloud cover percentage1:** specifies the median cloud cover percentage to move to for the CDFSII based precip estimate. A typical value is 15.0.

**AGRMET median cloud cover percentage2:** specifies the median cloud cover percentage to move to for the CDFSII based precip estimate. A typical value is 0.60.

**AGRMET overcast percentage:** specifies the overcast percentage to move to for CDFSII based precipitation estimate. A typical value is 0.30.

**AGRMET 3hr maximum precip ceiling:** specifies the 3 hour maximum precip ceiling value. A typical value is 200.0.

#### 7.2.14 CMAP forcing specification

This segment of the config file specifies the CMAP precipitation data. See Section 5 for instructions on how to obtain these forcing data-sets.

**CMAP forcing directory:** specifies the location of the CMAP forcing files. **CMAP domain x-dimension size:** and **CMAP domain y-dimension size:** specify the native domain parameters of the CMAP forcing data. The map projection is specified in the driver modules defined for the CMAP routines.

#### 7.2.15 Huffman forcing specification

This segment of the config file specifies the Huffman precipitation data. See Section 5 for instructions on how to obtain these forcing data-sets.

**Huffman forcing directory:** specifies the location of the Huffman forcing files. **Huffman domain x-dimension size:** and **Huffman domain y-dimension size:** specify the native domain parameters of the Huffman forcing data. The map projection is specified in the driver modules defined for the Huffman routines.

#### 7.2.16 CMORPH forcing specification

This segment of the config file specifies the CMORPH precipitation data. See Section 5 for instructions on how to obtain these forcing data-sets.

**CMORPH forcing directory:** specifies the location of the CMORPH forcing files. **CMORPH domain x-dimension size:** and **CMORPH domain y-dimension**

**size:** specify the native domain parameters of the CMORPH forcing data. The map projection is specified in the driver modules defined for the CMORPH routines.

### 7.2.17 Noah LSM specification

This segment of the config file specifies the run-time options and data specific to the Noah land surface model. See Section 5 for instructions on how to obtain these parameter data-sets.

**NOAH model output interval:** defines the output interval for Noah, in seconds. The typical value used in the LIS runs is 3 hours (=10800)

**NOAH restart output interval:** defines the restart writing interval for Noah, in seconds. The typical value used in the LIS runs is 24 hours (=86400).

**NOAH restart file:** specifies the Noah active restart file.

**NOAH vegetation parameter table:** specifies the Noah static vegetation parameter table file.

**NOAH soil parameter table:** specifies the Noah soil parameter file.

**NOAH general parameter table:** specifies the Noah general parameter file.

**NOAH bottom temperature climatology interval:** specifies in months, the climatology interval of the TBOT files. 0 indicates that the files are static.

**NOAH number of vegetation parameters:** specifies the number of static vegetation parameters specified for each veg type.

**NOAH soils scheme:** specifies the soil mapping scheme used. (1-zobler, 2-STATSGO)

**NOAH number of soil classes:** specifies the number soil classes in the above mapping scheme (9-zobler, 19-STATSGO)

**NOAH number of soil layers:** specifies the number of soil layers. The typical value used in Noah is 4.

**NOAH observation height:** specifies the height of meteorological observations. The typical value used in Noah is 6m.

**NOAH initial soil moisture:** specifies the initial volumetric soil moisture used in the cold start runs.

**NOAH initial soil temperature:** specifies the initial skin temperature used in the cold start runs.

A nested run using Noah should specify options for each nest similar to the following example, which shows the configuration for a run using one subnest.

```
NOAH model output interval:          10800 10800 #in seconds
NOAH restart output interval:        86400 86400 #in seconds
NOAH restart file:                  "noah.rst" "noah.rst"
NOAH vegetation parameter table:    "./noah_parms/noah.vegparms.txt" "./noah_parms/noah.vegparms.txt"
NOAH soil parameter table:          "./noah_parms/noah.soilparms.txt" "./noah_parms/noah.soilparms.txt"
NOAH general parameter table:       "./noah_parms/GENPARM.UNIF.TBL" "./noah_parms/GENPARM.UNIF.TBL"
NOAH bottom temperature climatology interval: 0 0 # in months, 0-static
```

```

NOAH number of vegetation parameters: 7 7
NOAH soils scheme: 1 1 #1-zobler, 2-statsgo
NOAH number of soil classes: 9 9 #9 for zobler, 19 for statsgo
NOAH number of soil layers: 4 4
NOAH observation height: 6 6 #meters
NOAH initial soil moisture: 0.30 0.30 # % volumetric
NOAH initial soil temperature: 290 290 # Kelvin

```

### 7.2.18 CLM LSM specification

This segment of the config file specifies the run-time options and data specific to the CLM2 land surface model. See Section 5 for instructions on how to obtain these parameter data-sets.

**CLM model output interval:** defines the output interval for CLM, in seconds. The typical value used in the LIS runs is 3 hours (=10800)

**CLM restart output interval:** defines the restart writing interval for CLM, in seconds. The typical value used in the LIS runs is 24 hours (=86400).

**CLM restart file:** specifies the CLM active restart file.

**CLM vegetation parameter file:** specifies vegetation type parameters look-up table.

**CLM canopy height table:** specifies the canopy top and bottom heights (for each vegetation type) look-up table.

**CLM initial soil moisture:** specifies the initial soil moisture wetness used in the cold start runs.

**CLM initial soil temperature:** specifies the initial soil temperature used in the cold start runs.

**CLM initial snow mass:** specifies the initial snow mass used in the cold start runs.

### 7.2.19 Model output configuration

This segment of the config file allows the specification of the variables to be written to the model output. By specifying 1/0 (1-write, 0-do not write), the users can select from among the following list of ALMA variables.

The following options specify the energy balance variables

**Swnet:** Net Shortwave Radiation (W/m<sup>2</sup>)

**Lwnet:** Net Longwave Radiation (W/m<sup>2</sup>)

**Qle:** Latent Heat Flux (W/m<sup>2</sup>)

**Qh:** Sensible Heat Flux (W/m<sup>2</sup>)

**Qg:** Ground Heat Flux (W/m<sup>2</sup>)

**Qf:** Energy of fusion (W/m<sup>2</sup>)

**Qv:** Energy of sublimation (W/m<sup>2</sup>)

**Qtau:** Momentum flux (N/m<sup>2</sup>)

**DelSurfHeat:** Change in surface heat storage (J/m<sup>2</sup>)

**DelColdCont:** Change in snow cold content (J/m<sup>2</sup>)

The following options specify the water balance variables

**Snowf:** Snowfall rate (kg/m<sup>2</sup>s)  
**Rainf:** Rainfall rate (kg/m<sup>2</sup>s)  
**Evap:** Total Evapotranspiration (kg/m<sup>2</sup>s)  
**Qs:** Surface runoff (kg/m<sup>2</sup>s)  
**Qrec:** Recharge (kg/m<sup>2</sup>s)  
**Qsb:** Subsurface runoff (kg/m<sup>2</sup>s)  
**Qsm:** Snowmelt (kg/m<sup>2</sup>s)  
**Qst:** Snow throughfall (kg/m<sup>2</sup>s)  
**DelSoilMoist:** Change in soil moisture (kg/m<sup>2</sup>)  
**DelsWE:** Change in snow water equivalent (kg/m<sup>2</sup>)  
**DelSurfStor:** Change in surface water storage (kg/m<sup>2</sup>)  
**DelIntercept:** Change in interception storage (kg/m<sup>2</sup>)

The following options specify the surface state variables

**SnowT:** Snow surface temperature (K)  
**VegT:** Vegetation canopy temperature (K)  
**BareSoilT:** Temperature of bare soil (K)  
**AvgSurfT:** Average surface temperature (K)  
**RadT:** Surface Radiative Temperature (K)  
**Albedo:** Surface Albedo (-)  
**SWE:** Snow Water Equivalent (kg/m<sup>2</sup>)  
**SWEVeg:** SWE intercepted by vegetation (kg/m<sup>2</sup>)  
**SurfStor:** Surface water storage (kg/m<sup>2</sup>)

The following options specify the sub surface state variables

**SoilMoist:** Average layer soil moisture (kg/m<sup>2</sup>)  
**SoilTemp:** Average layer soil temperature (K)  
**SmLiqFrac:** Average layer fraction of liquid moisture (-)  
**SmFrozFrac:** Average layer fraction of frozen moisture (-)  
**SoilWet:** Total soil wetness (-)

The following options specify the evaporation components

**PotEvap:** Potential Evapotranspiration (kg/m<sup>2</sup>s)  
**ECanop:** Interception evaporation (kg/m<sup>2</sup>s)  
**TVeg:** Vegetation transpiration (kg/m<sup>2</sup>s)  
**ESoil:** Bare soil evaporation (kg/m<sup>2</sup>s)  
**EWater:** Open water evaporation (kg/m<sup>2</sup>s)  
**RootMoist:** Root zone soil moisture (kg/m<sup>2</sup>)  
**CanopInt:** Total canopy water storage (kg/m<sup>2</sup>)  
**EvapSnow:** Snow evaporation (kg/m<sup>2</sup>s)  
**SubSnow:** Snow sublimation (kg/m<sup>2</sup>s)  
**SubSurf:** Sublimation of the snow free area (kg/m<sup>2</sup>s)  
**ACond:** Aerodynamic conductance

## 8 Output Data Processing

This section describes how to process the generated output.

The output data-sets created by running the LIS executable are written into sub-directories of the `$WORKING/LIS/OUTPUT/` directory (created at run-time). These sub-directories are named such as `EXP999`.

The output data consists of ASCII text files and model output in binary format.

For example, assume that you performed a “1/4 deg Noah with GEOS forcing” simulation for 11 June 2001, with an experiment code value of 999, and writing output as a 2-D array.

This run will produce a `$WORKING/LIS/OUTPUT/EXP999/` directory. This directory will contain:

File Name	Synopsis
Noahstats.dat	Statistical summary of output
NOAH	Directory containing output data

The `NOAH` directory will contain sub-directories of the form `YYYY/YYYYMMDD`, where `YYYY` is a 4-digit year and `YYYYMMDD` is a date written as a 4-digit year, 2-digit month and a 2-digit day; both corresponding to the running dates of the simulation.

For this example, `NOAH` will contain a `2001/20010611` sub-directory.

Its contents are the output files generated by the executable. They are:

```
2001061100.gs4r
2001061103.gs4r
2001061106.gs4r
2001061109.gs4r
2001061112.gs4r
2001061115.gs4r
2001061118.gs4r
2001061121.gs4r
```

Note, each file-name contains a date-stamp marking the year, month, day, and hour that the data corresponds to. The output data files for CLM and VIC are similar.

The generated output can be written in a 2-D grid format or as a 1-d vector. See Section 7.1.3 for more details. If written as a 1-d vector, the output must be converted into a 2-d grid before it can be visualized.

## A LIS config File

This is a sample LIS config file. The annotations embedded in the file explains various options.

```
#Overall driver options
Running mode:      1 # 1-retrospective , 2-AFWA AGRMET mode
Domain type:      6 # 1-latlon, 2-GSWP, 3-polar, 4-lambert
                           # 5-MERC, 6-AGRMET
Number of subnests:    0 # 0 - 1 domain, 1 - one nest
Land surface model:  1 # 1-noah, 2-CLM, 3-VIC, 4-Hyssib
Base forcing source: 7 # 1-GDAS, 2-GEOS,3-ECMWF,
                           # 4-NLDAS,5-GSWP,6-BERG, 7-AGRMET
Supplemental forcing source: 0 # 0-none

#The following options list the choice of parameter maps to be
#used
Landcover data source:      2 # 1-UMD, 2-USGS
Use soil texture:           1 # 1-use, 0-do not use
Soil data source:           2 # 1-FAO, 2-STATSGO
Soil color data source:     0 # 0-do not use, 1-FAO, 2-STATSGO
Topography data source:     0 # Not implemented yet
LAI data source:             0 # 0-do not use, 2-AVHRR, 3-MODIS
Albedo data source:          1 # 0-do not use, 1-NCEP
Greenness data source:       1 # 0-do not use, 1-NCEP
Porosity data source:        0 # 0-do not use
Ksat data source:            0 # 0-do not use
B parameter data source:    0 # 0-do not use
Quartz data source:          0 # 0-do not use
Snow data source:            1 # 0-do not use

#Runtime options
Experiment code:            '777' #max 3 character experiment code
Number of veg types:         24 #for USGS
Number of forcing variables: 10
Use elevation correction:    0 #0- do not use, 1-use lapse rate
Spatial interpolation method: 1 #1-bilinear, 2-conservative
Temporal interpolation method: 2 #1-linear,2-ubernext
Output forcing:              1 #0- no, 1-yes
Output methodology:          2 #0- no output,1-tiled, 2-gridded
Output data format:          1 #1-binary,2-grib,3-netcdf
Logging level:                1 #1-basic, 2-detailed, 3-debug
Start mode:                  1 #1-restart,2-coldstart
Starting year:                2005
Starting month:                 12
Starting day:                  01
```

```

Starting hour: 12
Starting minute: 0
Starting second: 0
Ending year: 2006
Ending month: 3
Ending day: 1
Ending hour: 12
Ending minute: 0
Ending second: 0
Model timestep: 900 1800 #of the outer nest, in seconds
Undefined value: -9999
Output directory: 'OUTPUT' #what else?
Diagnostic output file: 'lisdiag'
Number of ensembles per grid: 1

#The following options are used for subgrid tiling based on vegetation
Maximum number of tiles per grid: 1
Cutoff percentage: 0.05

#Processor Layout
#Should match the total number of processors used

Number of processors along x: 2
Number of processors along y: 2

#Data Assimilation Options

Assimilation algorithm: 0 #0-none,1-direct insertion,
#2-EKF,3-EnKF
Variable being assimilated: 1 #1-Soil moisture
Observation data source: 1 #0-dummy data, 1-TMI

#Ensemble Propagation options

Ensemble propagation algorithm: 0 # 0-none, 1-Rolf's

-----DOMAIN SPECIFICATION-----
#Definition of Running Domain
#Specify the domain extremes in latitude and longitude

run domain lower left lat: -59.75
run domain lower left lon: -179.75
run domain upper right lat: 89.75
run domain upper right lon: 179.75
run domain resolution dx: 0.5
run domain resolution dy: 0.5

```

```

#Definition of Parameter Domain

param domain lower left lat:           -59.75
param domain lower left lon:           -179.75
param domain upper right lat:          89.75
param domain upper right lon:          179.75
param domain resolution dx:            0.5
param domain resolution dy:            0.5

#-----PARAMETERS-----
#Metadata for Parameter maps
#Landcover and Landmask

landmask file:                      ./AFWA-1deg/landmask.1gd4r
landcover file:                     ./AFWA-1deg/USGSveg.1gd4r
landcover lower left lat:           -59.75
landcover lower left lon:           -179.75
landcover upper right lat:          89.75
landcover upper right lon:          179.75
landcover resolution (dx):          0.5
landcover resolution (dy):          0.5

#Topography maps
elevation map:                      .
slope map:                           ./AFWA-1deg/slopetype.1gd4r
aspect map:
curvature map:
topography lower left lat:          -59.75
topography lower left lon:          -179.75
topography upper right lat:         89.75
topography upper right lon:         179.75
topography resolution (dx):          0.5
topography resolution (dy):          0.5

#Soils maps
#saturated matric potential - psisat
#saturated hydraulic conductivity - ksat
soil texture map:                   ./AFWA-1deg/soiltype.1gd4r
sand fraction map:
clay fraction map:
silt fraction map:
soil color map:
porosity layer1 map:
porosity layer2 map:

```

```

porosity layer3 map:
saturated matric potential map:
saturated hydraulic conductivity map:
b parameter map:
quartz map:
soils lower left lat:           -59.75
soils lower left lon:           -179.75
soils upper right lat:          89.75
soils upper right lon:          179.75
soils resolution (dx):          0.5
soils resolution (dy):          0.5

#Albedo maps
albedo map:                   ./AFWA-1deg/alb
albedo climatology interval: 3 #in months
max snow free albedo map:    ./AFWA-1deg/mxsnoalb.1gd4r
bottom temperature map:      ./AFWA-1deg/tbot.1gd4r

#Greenness fraction maps
greenness fraction map:       ./AFWA-1deg/green
greenness climatology interval: 1 #in months

#LAI maps
LAI map:                      ./UMD-25KM

#snow depth maps
Snow depth map:               ./FORCING/AFWA/
-----FORCINGS-----
#GDAS (forcing option =1)
GDAS forcing directory:        ./FORCING/GDAS
GDAS elevation map:            ./UMD-25KM/gdas_elev.1gd4r
GDAS domain x-dimension size: 512 512
GDAS domain y-dimension size: 256 256
GDAS number of forcing variables: 15 15

#GEOS (forcing option =2)
GEOS forcing directory:        "" ""
GEOS domain x-dimension size: 360 360
GEOS domain y-dimension size: 181 181
GEOS number of forcing variables: 13 13

#ECMWF (forcing option =3)
ECMWF forcing directory:       "" ""
ECMWF domain x-dimension size: 1440 1440
ECMWF domain y-dimension size: 601 601

```

```

ECMWF number of forcing variables:      13    13

#NLDAS (forcing option =4)
NLDAS forcing directory:          ./FORCING/NLDAS
NLDAS elevation map:             ./UMD-25KM/edas_elev.1gd4r
NLDAS domain x-dimension size:   464   464
NLDAS domain y-dimension size:   224   224

#GSWP (forcing option =5)
GSWP 2m air temperature map:     "./gswp2data/Tair_cru/Tair_cru"
GSWP 2m specific humidity map:   "./gswp2data/Qair_cru/Qair_cru"
GSWP wind map:                  "./gswp2data/Wind_ncep/Wind_ncep"
GSWP surface pressure map:       "./gswp2data/PSurf_ecor/PSurf_ecor"
GSWP rainfall rate map:         "./gswp2data/Rainf_gswp/Rainf_gswp"
GSWP snowfall rate map:          "./gswp2data/Snowf_gswp/Snowf_gswp"
GSWP incident shortwave radiation map: "./gswp2data/SWdown_srb/SWdown_srb"
GSWP incident longwave radiation map: "./gswp2data/LWdown_srb/LWdown_srb"

#NCEP/ECMWF reanalysis (forcing option =6)
BERG forcing directory:          "./FORCING/GRID"
BERG maskfile:                  "./UMD-3KM/ecmwf_land_sea.05"
BERG elevation map:              "./UMD-3KM/berg_elev.1gd4r"
BERG domain x-dimension size:   720
BERG domain y-dimension size:   360

#AFWA/AGRMET (forcing option =7)
AGRMET forcing directory:        ./FORCING/AFWA1
AGRMET analysis directory:       ./Analysis
AGRMET polar mask file:         ./PARAMS/AFWA/MESH8/point_switches
AGRMET polar terrain file:      ./PARAMS/AFWA/MESH8/terrain
AGRMET sfcalc cntm file:        ./PARAMS/AFWA/MESH8/sfcalc-cntm
AGRMET precip climatology:      ./PARAMS/AFWA/MESH8/pcp_clim/
AGRMET nogaps wind weight:      0.75
AGRMET minimum wind speed:      0.25
AGRMET use present/past weather estimate: 1
AGRMET use precip observations: 1
AGRMET use SSMI data:           1
AGRMET use CDFSII-based estimate: 1
AGRMET use GEOPRECIP estimate: 2
AGRMET CDFSII time interval:   6
AGRMET use precip climatology: 1
AGRMET SSMI zero use switch:    1
AGRMET snow distribution shape parameter: 2.6
AGRMET alternate monthly weighting factor: 1.0
AGRMET minimum 3hr climo value: 0.025
AGRMET maximum 3hr climo value: 0.375

```

```

AGRMET minimum precip-per-precip day multiplier: 0.0
AGRMET maximum precip-per-precip day multiplier: 1.1
AGRMET cloud threshold to generate CDFSII estimate: 85.0
AGRMET median cloud cover percentage1: 15.0
AGRMET median cloud cover percentage2: 0.60
AGRMET overcast percentage: 0.30
AGRMET 3hr maximum precip ceiling: 200.0

#CPC CMAP precipitation (observed forcing option = 2)
CMAP forcing directory: ""
CMAP domain x-dimension size: 512
CMAP domain y-dimension size: 256

#HUFFMAN precipitation (observed forcing option = ?)
Huffman forcing directory: ""
Huffman domain x-dimension size: 1440
Huffman domain y-dimension size: 480

#HUFFMAN Version 6 precipitation (observed forcing option = ?)
Huffman V6 forcing directory: ""
Huffman V6 domain x-dimension size: 1440
Huffman V6 domain y-dimension size: 400

#CMORPH 30min, 8KM (observed forcing option = ?)
CMORPH forcing directory: ""
CMORPH domain x-dimension size: 4948
CMORPH domain y-dimension size: 1649

#CMORPH 3hrly, 25KM (observed forcing option = ?)
CMORPH 25KM forcing directory: ""
CMORPH 25KM domain x-dimension size: 1440
CMORPH 25KM domain y-dimension size: 480

#CEOP Station data (observed forcing option=?)
CEOP location index: 4
CEOP data directory: ""
CEOP metadata file: ""

-----LAND SURFACE MODELS-----
#NCEP's NOAH (lsm option =1)
NOAH model output interval: 10800 #in seconds
NOAH restart output interval: 86400 #in seconds
NOAH restart file: "noah.rst"
NOAH slope file: "./AFWA-1deg/slopetype.1gd4r"
NOAH vegetation parameter table: "./noah_parms/noah.vegparms.afwa"
NOAH soil parameter table: "./noah_parms/noah.soilparms.afwa"

```

```

NOAH general parameter table:      "./noah_parms/GENPARM.UNIF.TBL"
NOAH bottom temperature climatology interval: 0 # in months, 0-static
NOAH number of vegetation parameters: 7
NOAH soils scheme:                2 #1-zobler, 2-statsgo
NOAH number of soil classes:      16 #9 for zobler, 19 for statsgo, 16 afw
NOAH number of soil layers:       4
NOAH observation height:         20 #meters
NOAH initial soil moisture:      0.30 # % volumetric
NOAH initial soil temperature:   290 # Kelvin

#NCAR's CLM2.0 (lsm option =2)
CLM model output interval:       10800
CLM restart output interval:     86400
CLM restart file:               "clm2.rst"
CLM vegetation parameter table:  ./clm_parms/umdvegparam.txt
CLM canopy height table:         ./clm_parms/clm2_ptcanhts.txt
CLM initial soil moisture:       0.45 # % volumetric
CLM initial soil temperature:   290.0 # Kelvin
CLM initial snow mass:          0.0

#VIC from princeton university (lsm option = 3, not supported in this release)
# not yet supported in this version
VIC model output interval:        900
VIC restart output interval:      86400
VIC number of soil layers:        3
VIC number of soil thermal nodes: 5
VIC number of snow bands:         1
VIC number of rootzones:          2
VIC soil parameter file:          ""
VIC vegetation parameter table:   ""
VIC restart file:                 ""
VIC Ds map:                      ""
VIC DSMax map:                   ""
VIC Ws map:                      ""
VIC infiltration capacity map:   ""
VIC depth1 map:                  ""
VIC depth2 map:                  ""
VIC depth3 map:                  ""

#Mosaic from NASA GSFC (lsm option =4)
Mosaic model output interval:    900
Mosaic restart output interval:  86400
Mosaic restart file:             "mos.rst"
Mosaic vegetation parameter table: "./mos_parms/real.vegiparms.txt"
Mosaic monthly vegetation parameter table: "./mos_parms/real.monvegpar.txt"
Mosaic soil parameter table:     "./mos_parms/real.soilparms.txt"

```

```

Mosaic initial soil moisture:          0.3
Mosaic initial soil temperature:       290.0

#HySSIB from NASA GSFC (lsm option =5, not yet supported in this version)
HySSIB model output interval:        900

#SSIB (lsm option =6, not yet supported in this version)

#Catchment (lsm option=7,not yet supported in this version)

#-----MODEL OUTPUT CONFIGURATION-----
#Specify the list of ALMA variables that need to be featured in the
#LSM model output

#Energy balance components
Swnet:      1      # Net Shortwave Radiation (W/m2)
Lwnet:      1      # Net Longwave Radiation (W/m2)
Qle:        1      # Latent Heat Flux (W/m2)
Qh:         1      # Sensible Heat Flux (W/m2)
Qg:         1      # Ground Heat Flux (W/m2)
Qf:         0      # Energy of fusion (W/m2)
Qv:         0      # Energy of sublimation (W/m2)
Qtau:       0      # Momentum flux (N/m2)
DelSurfHeat: 0      # Change in surface heat storage (J/m2)
DelColdCont: 0      # Change in snow cold content (J/m2)

#Water balance components
Snowf:      0      # Snowfall rate (kg/m2s)
Rainf:      0      # Rainfall rate (kg/m2s)
Evap:       0      # Total Evapotranspiration (kg/m2s)
Qs:         1      # Surface runoff (kg/m2s)
Qrec:       0      # Recharge (kg/m2s)
Qsb:        1      # Subsurface runoff (kg/m2s)
Qsm:        0      # Snowmelt (kg/m2s)
Qst:        0      # Snow throughfall (kg/m2s)
DelSoilMoist: 0     # Change in soil moisture (kg/m2)
DelsWE:     0      # Change in snow water equivalent (kg/m2)
DelSurfStor: 0     # Change in surface water storage (kg/m2)
DelIntercept: 0    # Change in interception storage (kg/m2)

#Surface State Variables
SnowT:      0      # Snow surface temperature (K)
VegT:       0      # Vegetation canopy temperature (K)
BareSoilT:   0      # Temperature of bare soil (K)
AvgSurfT:   1      # Average surface temperature (K)
RadT:       0      # Surface Radiative Temperature (K)

```

```

Albedo:      1      # Surface Albedo (-)
SWE:        1      # Snow Water Equivalent (kg/m2)
SWEVeg:     0      # SWE intercepted by vegetation (kg/m2)
SurfStor:   0      # Surface water storage (kg/m2)

#Subsurface State Variables
SoilMoist:   1      # Average layer soil moisture (kg/m2)
SoilTemp:    1      # Average layer soil temperature (K)
SmLiqFrac:   0      # Average layer fraction of liquid moisture (-)
SmFrozFrac:  0      # Average layer fraction of frozen moisture (-)
SoilWet:     0      # Total soil wetness (-)

#Evaporation components
PotEvap:    0      # Potential Evapotranspiration (kg/m2s)
ECanop:     0      # Interception evaporation (kg/m2s)
TVeg:       0      # Vegetation transpiration (kg/m2s)
ESoil:      0      # Bare soil evaporation (kg/m2s)
EWater:     0      # Open water evaporation (kg/m2s)
RootMoist:   0      # Root zone soil moisture (kg/m2)
CanopInt:   1      # Total canopy water storage (kg/m2)
EvapSnow:    0      # Snow evaporation (kg/m2s)
SubSnow:    0      # Snow sublimation (kg/m2s)
SubSurf:    0      # Sublimation of the snow free area (kg/m2s)
ACond:      0      # Aerodynamic conductance

```

## B Makefile

```
# Set up special characters

null  :=
space := $(null) $(null)
doctool :=../../utils/docsgen.sh

# Check for directory in which to put executable
ifeq ($(MODEL_EXEDIR),$(null))
MODEL_EXEDIR := .
endif

# Check for name of executable
ifeq ($(EXENAME),$(null))
EXENAME := LIS
endif

# Check if SPMD is defined in "misc.h"
# Ensure that it is defined and not just "undef SPMD" set in file
ifeq ($(SPMD),$(null))
    SPMDSET := $(shell /bin/grep SPMD misc.h)
    ifneq (,$(findstring define,$(SPMDSET)))
        SPMD := TRUE
    else
        SPMD := FALSE
    endif
endif

# Determine platform
UNAMES := $(shell uname -s)
UMACHINE := $(shell uname -m)

ifeq ($(UNAMES),IRIX64)
INC_NETCDF := /u/jvg/local/netcdf-3.5.1/include
LIB_NETCDF := /u/jvg/local/netcdf-3.5.1/lib
ESMF_DIR    := ../../lib/esmf
LIB_ESMF    := $(ESMF_DIR)/lib/lib0
MOD_ESMF   := $(ESMF_DIR)/mod/mod0

endif

ifeq ($(UNAMES),AIX)
LIB_NETCDF := /usrx/local/netcdf
```

```

INC_NETCDF := /usrx/local/netcdf/include
LIB_MPI     := /usr/lpp/ppe.poe/lib
INC_MPI     := /usr/lpp/ppe.poe/include
ESMF_DIR    := /nfsuser/g01/wx20je/LDAS/src-par/esmf
LIB_ESMF    := $(ESMF_DIR)/lib/lib0
MOD_ESMF    := $(ESMF_DIR)/mod/mod0

endif

ifeq ($(UNAMES),OSF1)

INC_NETCDF := /usr/ulocal/include
LIB_NETCDF := /usr/ulocal/lib
LIB_MPI := /usr/lib
INC_MPI := /usr/include
ESMF_DIR := ../lib/esmf
LIB_ESMF := $(ESMF_DIR)/lib/lib0
MOD_ESMF := $(ESMF_DIR)/mod/mod0

endif

ifeq ($(UMACHINE), i686)

INC_NETCDF := /data1/netcdf/include
LIB_NETCDF := /data1/netcdf/lib
ESMF_DIR := ../lib/esmf
LIB_ESMF := $(ESMF_DIR)/lib/lib0
MOD_ESMF := $(ESMF_DIR)/mod/mod0
ifeq ($(LIS_ARCH), linux_absoft)
MPI_PREFIX := /data1/jim/local/mpich-1.2.4-absoft
endif
ifeq ($(LIS_ARCH), linux_pgi)
MPI_PREFIX := /data1/jim/local/mpich-1.2.6-pgi
endif
LIB_MPI := $(MPI_PREFIX)/lib
INC_MPI := $(MPI_PREFIX)/include

endif

# Load dependency search path.
dirs := . $(shell cat Filepath)
# Set cpp search path, include netcdf
ifeq ($(LIS_SPMD), single)
cpp_dirs := $(dirs)
else
cpp_dirs := $(dirs) $(INC_MPI)

```

```

endif
cpp_path := $(foreach dir,$(cpp_dirs),-I$(dir)) # format for command line

# Expand any tildes in directory names. Change spaces to colons.
VPATH      := $(foreach dir,$(cpp_dirs),$(wildcard $(dir)))
VPATH      := $(subst $(space),:,$(VPATH))

#-----
# Primary target: build the model
#-----
all: $(MODEL_EXEDIR)/$(EXENAME)

# Get list of files and determine objects and dependency files
FIND_FILES = $(wildcard $(dir)/*.F $(dir)/*.f $(dir)/*.F90 $(dir)/*.c $(dir)/*.f)
FILES      = $(foreach dir, $(dirs),$(FIND_FILES))
SOURCES    := $(sort $(notdir $(FILES)))
DEPS       := $(addsuffix .d, $(basename $(SOURCES)))
OBJS       := $(addsuffix .o, $(basename $(SOURCES)))
DOCS       := $(addsuffix .tex, $(basename $(SOURCES)))

$(MODEL_EXEDIR)/$(EXENAME): $(OBJS)
    $(FC) -o $@ $(OBJS) $(FOPTS) $(LDFLAGS)
debug: $(OBJS)
    echo "FFLAGS: $(FFLAGS)"
    echo "LDFLAGS: $(LDFLAGS)"
    echo "OBJS: $(OBJS)"

***** Architecture-specific flags and rules *****
***** SGI *****

ifeq ($(UNAMES),IRIX64)

ESMF_ARCH = IRIX64
FC        := f90
CPP       := /lib/cpp

# Library directories
LIB_DIR   = ./lib/sgi-64/
HDFLIBDIR = $(LIB_DIR)
GFIOLIBDIR = $(LIB_DIR)

```

```

CPPFLAGS      := -P
PSASINC       :=

# non-opendap
CFLAGS        := $(cpp_path) -64 -c -O2 -OPT:Olimit=0 -static -DIRIX64
FFLAGS        = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG -DNO

# opendap
#CFLAGS        := $(cpp_path) -64 -c -O2 -OPT:Olimit=0 -static -DIRIX64 -DOPENDAP
#FFLAGS        = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG -DNO

# debugging
#CFLAGS        := $(cpp_path) -64 -c -g -static -DIRIX64
#FFLAGS        = $(cpp_path) -64 -r4 -i4 -c -cpp -I$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG -DNO

FOPTS = $(LIB_DIR)bacio_64_sgi $(LIB_DIR)w3lib_64_sgi

LDFLAGS       = -64 -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -lmpi -L$(LIB_NETCDF) -lnetcdf

# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the -= -mp is all about)

ifeq ($(SPMD),TRUE)
  FFLAGS   -= -mp
  FFLAGS   += -macro_expand
#  FFLAGS   += -I$(INC_MPI) -macro_expand

#  LDFLAGS += -L$(LIB_MPI) -lmpi
else
  FFLAGS   += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .F90 .f .c .o

.F90.o:
$(FC) $(FFLAGS) $<
.f.o:
$(FC) $(FFLAGS) $<
.c.o:
cc $(cpp_path) $(CFLAGS) $<

endif

#-----

```

```

# AIX
#-----
ifeq ($(UNAMES),AIX)

ESMF_ARCH      = rs6000_64
#FC            = mpxlf90_r
FC             = mpxlf90
CPP            = /lib/cpp
#CC            = mpcc_r
CC             = mpcc
MPI_PATH       = /usr/lpp/ppe.poe

# Library directories
LIB_DIR        = /nwprod/lib/
LIB_DIR2       = ../lib/
HDFLIBDIR     = $(LIB_DIR)
GFIOLIBDIR    = $(LIB_DIR)
CPPFLAGS       = -P
PSASINC        =
CFLAGS          = $(cpp_path) -c -w -O -q64 -qcpluscmt -DSPMD
FFLAGS          = $(cpp_path) -I$(MPI_PATH)/include -O -qstrict -qsuffix=f=F90:cpp=F90 -qtune=a
FOPTS           =
LDFLAGS         = -q64 -bmap:map -bloadmap:lm -lmass ..../lib/w3lib/libw3.a ..../lib/read_grib/read
ifeq ($(SPMD),TRUE)
# FFLAGS  -= -mp
# FFLAGS  += -macro_expand
# FFLAGS  += -I$(INC_MPI) -macro_expand

# LDFLAGS += -L$(LIB_MPI) -lmpi
else
  FFLAGS  +=
endif
.F.o:
$(FC) $(FFLAGS) $<
%.o:%.F90
$(FC) $(FFLAGS) $<
.F90.o:
$(FC) $(FFLAGS) $<
.f.o:
$(FC) $(FFLAGS) $<
.f90.o:
.c.o:
$(CC) $(cpp_path) $(CFLAGS) $<

endif

```

```

#-----
# Compaq alpha - Halem cluster
#-----
ifeq ($(UNAMES),OSF1)
ifeq ($(LIS_SPMD),parallel)
ESMF_ARCH = alpha
FC        := f90
CPP       := /lib/cpp

# Library directories
LIB_DIR  = ../lib/w3lib/
CPPFLAGS := -P
PSASINC  :=
CFLAGS   := $(cpp_path) -g -n32 -DOSF1 -DSPMD
FFLAGS   = $(cpp_path) -c -g -cpp -I$(INC_NETCDF) -automatic -convert big_endian -assume
#FFLAGS    = $(cpp_path) -c -cpp -automatic -convert big_endian -assume byterecl -arch ev6
#      -check bounds -check format -warn argument_checking -warn declarations -warn noalignm
# -warn uninitialized -warn nousage -warn unused -fpe0\
FFLAGS_DOTF90 = -DHIDE_SHR_MSG -DOSF1 -free -fpe3 -DNO_SHR_VMATH
FFLAGS_DOTF = -extend_source -omp -automatic
FOPTS = $(LIB_DIR)libw3.a ../lib/read_grib/readgrib.a
LDFLAGS    = -lmpi -L$(LIB_NETCDF) -lnetcdf
else
ESMF_ARCH = alpha
FC        := f90
CPP       := /lib/cpp

# Library directories
LIB_DIR  = ../lib/w3lib/
CPPFLAGS := -P
PSASINC  :=
CFLAGS   := $(cpp_path) -n32 -DOSF1
FFLAGS   = $(cpp_path) -c -cpp -automatic -convert big_endian -assume byterecl -arch ev6
          -I$(MOD_ESMF)/$(ESMF_ARCH) -DOSF1 \
          -DHIDE_SHR_MSG -DNO_SHR_VMATH
FFLAGS_DOTF90 = -DHIDE_SHR_MSG -DOSF1 -free -fpe3 -DNO_SHR_VMATH
FFLAGS_DOTF = -extend_source -omp -automatic
FOPTS = $(LIB_DIR)libw3.a
LDFLAGS    = -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf
endif
# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the == -mp is all about)

ifeq ($(SPMD),TRUE)

```

```

FFLAGS -= -mp
FFLAGS += -I$(INC_MPI) -macro_expand

# LDFLAGS += -L$(LIB_MPI) -lmpi
else
    FFLAGS += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .f .f90 .F90 .c .o

.f.o:
$(FC) $(FFLAGS) $<
.F90.o:
$(FC) $(FFLAGS) $<
.c.o:
cc -c $(cpp_path) $(CFLAGS) $<
.f90.o:
$(FC) $(FFLAGS) $<

endif
#-----
# Linux
#-----

ifeq ($(UMACHINE),i686)

ifeq ($(LIS_ARCH),linux_absoft)
ifeq ($(LIS_SPMD),single)
FC := f90
CC := gcc
else
FC := $(MPI_PREFIX)/bin/mpif90
CC := $(MPI_PREFIX)/bin/mpicc
endif
CPP := /lib/cpp
ifeq ($(LIS_SPMD),single)
CFLAGS := $(cpp_path) -Wall -c -O2 -DABSOFT -DLITTLE_ENDIAN
FFLAGS := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 -YDEALLOC=ALL -DHIDE_SH
else
CFLAGS := $(cpp_path) -Wall -c -O2 -DSPMD -DABSOFT -DLITTLE_ENDIAN
FFLAGS := $(cpp_path) -c -O2 -YEXT_NAMES=LCS -s -B108 -YCFRL=1 -YDEALLOC=ALL -DSPMD -D
endif
ifeq ($(LIS_SPMD),single)
LDFLAGS := -LU77 -lm

```

```

else
LDFLAGS      := -lmpich -lU77 -lm
endif
endif

ifeq ($(LIS_ARCH),linux_pgi)
ifeq ($(LIS_SPMD),single)
FC          := pgf90
CC          := gcc
else
FC          := $(MPI_PREFIX)/bin/mpif90
CC          := $(MPI_PREFIX)/bin/mpicc
endif
CPP         := /lib/cpp
ifeq ($(LIS_SPMD),single)
CFLAGS      := $(cpp_path) -c -O2 -DPGI
FFLAGS      := $(cpp_path) -c -O2 -DHIDE_SHR_MSG -DNO_SHR_VMATH -DPGI -Mbyteswapi -r4 -i4 -
else
CFLAGS      := $(cpp_path) -c -O2 -DSPMD -DPGI
FFLAGS      := $(cpp_path) -c -O2 -DSPMD -DHIDE_SHR_MSG -DNO_SHR_VMATH -DPGI -Mbyteswapi -r4 -i4 -
endif
ifeq ($(LIS_SPMD),single)
LDFLAGS     := -lm -Mlf
else
LDFLAGS     := -lmpich -lm -Mlf
endif
endif

ifeq ($(LIS_ARCH),linux_lf95)
ifeq ($(LIS_SPMD), single)
FC          := lf95
endif
CPP         := /lib/cpp
CFLAGS      := $(cpp_path) -c -O -DUSE_GCC -DLAHEY -DLITTLE_ENDIAN
FFLAGS      := $(cpp_path) -c -O -DHIDE_SHR_MSG -DLINUX -DNO_SHR_VMATH -I$(MOD_ESMF)/$(ESMF_-
LDFLAGS     := -s --staticlink

endif

ifeq ($(LIS_ARCH),linux_ifc)

FC          := $(MPI_PREFIX)/bin/mpif90
CPP         := /lib/cpp

CFLAGS      := $(cpp_path) -c -O2
FFLAGS      := $(cpp_path) -c -I$(MOD_ESMF)/$(ESMF_ARCH) -DHIDE_SHR_MSG -DNO_SHR_VMATH -O

```

```

LDFLAGS      = -L$(LIB_ESMF)/$(ESMF_ARCH) -lesmf -lmpich
endif

# Library directories
#LIB_DIR   = ./lib/pc-32/$(ESMF_ARCH)/
LIB_DIR    = ./lib/w3lib/
HDFLIBDIR  = $(LIB_DIR)
GFIOLIBDIR = $(LIB_DIR)
CPPFLAGS    := -P
PSASINC     :=
#FOPTS = $(LIB_DIR)bacio_32_pclinux $(LIB_DIR)w3lib_32_pclinux
FOPTS = $(LIB_DIR)libw3.a
# WARNING: -mp and -g together cause wrong answers

# WARNING: - Don't run hybrid on SGI (that's what the -= -mp is all about)

ifeq ($(SPMD),TRUE)
# FFLAGS -= -mp
# FFLAGS += -macro_expand
# FFLAGS += -I$(INC_MPI) -macro_expand

# LDFLAGS += -L$(LIB_MPI) -lmpi
else
    FFLAGS += -DHIDE_MPI
endif

.SUFFIXES:
.SUFFIXES: .F90 .c .o

.F90.o:
$(FC) $(FFLAGS) $<
.c.o:
$(CC) $(cpp_path) $(CFLAGS) $<

endif

RM := rm
# Add user defined compiler flags if set, and replace FC if USER option set.
FFLAGS += $(USER_FFLAGS)
ifneq ($(USER_FC),$(null))
FC := $(USER_FC)
endif

clean:

```

```

$(RM) -f *.o *.mod *.stb $(MODEL_EXEDIR)/$(EXENAME)

realclean:
$(RM) -f *.o *.d *.mod *.stb $(MODEL_EXEDIR)/$(EXENAME)
doc:
$(doctool)
#-----
#!!!!!!!!!!!!!!DO NOT EDIT BELOW THIS LINE.!!!!!!!!!!!!!!
#-----
# These rules cause a dependency file to be generated for each source
# file. It is assumed that the tool "makdep" (provided with this
# distribution in clm2/tools/makdep) has been built and is available in
# the user's $PATH. Files contained in the clm2 distribution are the
# only files which are considered in generating each dependency. The
# following filters are applied to exclude any files which are not in
# the distribution (e.g. system header files like stdio.h).
#
# 1) Remove full paths from dependencies. This means gnumake will not break
#     if new versions of files are created in the directory hierarchy
#     specified by VPATH.
#
# 2) Because of 1) above, remove any file dependencies for files not in the
#     clm2 source distribution.
#
# Finally, add the dependency file as a target of the dependency rules. This
# is done so that the dependency file will automatically be regenerated
# when necessary.
#
#     i.e. change rule
#         make.o : make.c make.h
#         to:
#         make.o make.d : make.c make.h
#-----
DEPGEN := ./MAKDEP/makdep -s F
%.d : %.c
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
%.d : %.f
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
%.d : %.F90
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@
%.d : %.F
@echo "Building dependency file $@"
@$(DEPGEN) -f $(cpp_path) $< > $@

```

```
#  
# if goal is clean or realclean then don't include .d files  
# without this is a hack, missing dependency files will be created  
# and then deleted as part of the cleaning process  
#  
INCLUDE_DEPS=TRUE  
ifeq ($(MAKECMDGOALS), realclean)  
    INCLUDE_DEPS=FALSE  
endif  
ifeq ($(MAKECMDGOALS), clean)  
    INCLUDE_DEPS=FALSE  
endif  
  
ifeq ($(INCLUDE_DEPS), TRUE)  
-include $(DEPS)  
endif
```

## C READ GRIB - Information and Instructions

Thanks to Kristi Arsenault for putting this together

### **Caveat:**

This is a package of subroutines to read GRIB-formatted data. It is still under continuous development. It won't be able to read every GRIB dataset you give it, but it will read a good many.

- Kevin W. Manning  
NCAR/MMM  
Summer 1998, and continuing

The main user interfaces are:

**SUBROUTINE GRIBGET(NUNIT, IERR)** - Read a single GRIB record from UNIX file-descriptor NUNIT into array GREC. No unpacking of any header or data values is performed.

**SUBROUTINE GRIBREAD(NUNIT, DATA, NDATA, IERR)** - Read a single GRIB record from UNIX file-descriptor NUNIT, and unpack all header and data values into the appropriate arrays.

**SUBROUTINE GRIBHEADER(IERR)** - Unpack the header of a GRIB record

**SUBROUTINE GRIBDATA(DATARRAY, NDAT)** - Unpack the data in a GRIB record into array DATARRAY

**SUBROUTINE GRIBPRINT(ISEC)** - Print the header information from GRIB section ISEC.

**SUBROUTINE GET\_SEC1(KSEC1)** - Return the header information from Section 1.

**SUBROUTINE GET\_SEC2(KSEC2)** - Return the header information from Section 2.

**SUBROUTINE GET\_GRIDINFO(IGINFO, GINFO)** - Return the grid information of the previously-unpacked GRIB header.

**C-ROUTINE – COPEN(UNIT, NUNIT, NAME, MODE, ERR, OFLAG)** - Opens the GRIB file for reading later by the other routines.

### C.1 SUBROUTINE GRIBGET (NUNIT, IERR)

- Read a single GRIB record from UNIX file-descriptor NUNIT into array GREC. No unpacking of any header or data values is performed.

- NOTE!: Intrinsic parameter, ied, identifies type of GRIB edition of GRIB file trying to open. Below are the codes (also see Table1):

- If IED == 1, then GRIB Edition 1 has the size of the whole GRIB record right up front.
- If IED == 0, then GRIB Edition 0 does not include the total size, so we have to sum up the sizes of the individual sections

- If IED > 1, then STOP, if not GRIB Edition 0 or 1

Input: NUNIT (integer): C unit number to read from. This should already be open.

Output: IERR (integer): Error flag, Non-zero means there was a problem with the read.

Side Effects: The array GREC is allocated, and filled with one GRIB record. The C unit pointer is moved to the end of the GRIB record just read.

## C.2 SUBROUTINE GRIBREAD (NUNIT, DATA, NDATA, IERR)

- Read a single GRIB record from UNIX file-descriptor, NUNIT, unpack all header and data values into the appropriate arrays, and fill the allocatable array, DATARRAY(:).

Input: NUNIT (integer): C Unit to read from.

NDATA (integer): Size of array DATA (Should be  $\leq$  NDAT as computed herein.)

Output: DATA (real): The unpacked data array (dimension size of NDATA)  
IERR(integer): Error flag, non-zero means there was a problem.

Side Effects: Header arrays SEC0, SEC1, SEC2, SEC3, SEC4, XEC4, INFOGRID and INFOGRID are filled.

The BITMAP array is filled.

The C unit pointer is advanced to the end of the GRIB record.

## C.3 SUBROUTINE GRIBHEADER (IERR)

Unpack the header of a GRIB record

IERR non-zero means there was a problem unpacking the grib header

IERR (integer)

## C.4 SUBROUTINE GRIBDATA (DATARRAY, NDAT)

- Read and unpack the data in a GRIB record into array, DATARRAY

Input: NDAT (integer): The size of the data array we expect to unpack.

Output: DATARRAY (real): The unpacked data from the GRIB record (dimension size of NDAT)

Side Effects: - STOP – if it cannot unpack the data.

## C.5 SUBROUTINE GRIBPRINT (ISEC)

Print the header information from GRIB section ISEC.

ISEC Information can be found in the section C.10

## C.6 SUBROUTINE GET\_SEC1 (KSEC1)

- Return the header information from GRIB Section 1 (see C.10, TABLE 2).
- Return the GRIB Section 1 header information, which has already been unpacked by subroutine GRIBHEADER.
- KSEC1 (integer :: dimension(100))

## C.7 SUBROUTINE GET\_SEC2 (KSEC2)

- Return the header information from GRIB Section 2 (see C.10, TABLE 3).
- Return the GRIB Section 2 header information, which has already been unpacked by subroutine GRIBHEADER.
- KSEC2 (integer :: dimension(10))

## C.8 SUBROUTINE GET\_GRIDINFO (IGINFO, GINFO)

- Return the grid information of the previously-unpacked GRIB header.
  - IGINFO (integer :: dimension(40))
  - GINFO (real :: dimension(40))
- \*\* NOTE IGINFO and GINFO contain equivalent information, except that IGINFO is the integer form and GINFO is the real form.

## C.9 C-ROUTINE COPEN (UNIT, NUNIT, NAME, MODE, ERR, OFLAG)

- Opens the GRIB file for reading later by the other subroutines

UNIT = Fortran unit number (integer)

NUNIT = UNIX file descriptor associated with 'unit' (integer)

NAME = UNIX file name (character (len=120) )

MODE = 0 : write only - file will be created if it doesn't exist, - otherwise  
will be rewritten (integer)  
= 1 : read only

= 2 : read/write

ERR = 0 : no error opening file (integer)  
 $\neq 0$  : Error opening file

OFLAG = 0 : file name printed (no errors printed) (integer)  
 $> 0$  : file name printed and errors are printed  
 $< 0$  : no print at all (not even errors)

## C.10 SEC Header Array Information Tables

Please refer to <http://www.wmo.ch/web/www/WDM/Guides/Guide-binary-2.html> for additional GRIB1 header information

Table 3: SEC2: GRIB Header Section 2 information

Octet	GDS Content	
1-3	Length of GRIB Section 2 (in octets)	
4	Number of vertical-coordinate parameters	
5	Starting-point of the list of vertical-coordinate parameters	
6	Data-representation type (i.e., grid type) See GRIB Table 6 0 = Latitude/Longitude grid 3 = Lambert-conformal grid. 5 = Polar-stereographic grid.	
if(sec2(4)==0) then Lat/lon grid		
INFOGRID	Octet	GDS Content
1	7-8	I Dimension of the grid
2	9-10	J Dimension of the grid
3	11-13	Starting Latitude of the grid.
4	14-16	Starting Longitude of the grid.
5	17	Resolution and component flags.
6	18-20	Ending latitude of the grid.
7	21-23	Ending longitude of the grid.
8	24-25	Longitudinal increment.
9	26-27	Latitudinal increment.
10	28	Scanning mode (bit 3 from Table 8)
21	28	Iscan sign (+1/-1) (bit 1 from Table 8)
22	28	Jscan sign (+1/-1) (bit 2 from Table 8)
if(sec2(4)==1) then mercator grid		
INFOGRID	Octet	GDS Content
1	7-8	I Dimension of the grid

2	9-10	J Dimension of the grid
3	11-13	Starting Latitude of the grid.
4	14-16	Starting Longitude of the grid.
5	17	Resolution and component flags.
6	18-20	Ending latitude of the grid.
7	21-23	Ending longitude of the grid.
8	24-26	LATIN- The latitude(s) at which Mercator projection cylinder intersects the earth
9	27	Reserved (set to 0)
10	28	Scanning mode (bit 3 from Table 8)
11		True Lat
21	29-31	Iscan sign (+1/-1) (bit 1 from Table 8)
22	32-34	Jscan sign (+1/-1) (bit 2 from Table 8)
if(sec2(4)==3) then Lambert Conformal grid		
INFOGRID	Octet	GDS Content
1	7-8	I Dimension of the grid
2	9-10	J Dimension of the grid
3	11-13	Starting Latitude of the grid.
4	14-16	Starting Longitude of the grid.
5	17	Resolution and component flags.
6	18-20	Center longitude of the projection.
7	21-23	Grid-spacing in the I direction
8	24-26	Grid-spacing in the J direction
9	27	Projection center
10	28	Scanning mode (bit 3 from Table 8)
11	29-31	First TRUELAT value.
12	32-34	Second TRUELAT value.
13	35-37	Latitude of the southern pole
14	38-40	Longitude of the southern pole
21	41	Iscan sign (+1/-1) (bit 1 from Table 8)
22	42	Jscan sign (+1/-1) (bit 2 from Table 8)
if(sec2(4)==4) then Gaussian grid		
INFOGRID	Octet	GDS Content
1	7-8	I Dimension of the grid
2	9-10	J Dimension of the grid
3	11-13	Starting Latitude of the grid.
4	14-16	Starting Longitude of the grid.
5	17	Resolution and component flags.
6	18-20	Ending latitude of the grid.
7	21-23	Ending longitude of the grid.
8	24-25	Longitudinal increment.
9	26-27	Latitudinal increment.
10	28	Scanning mode (bit 3 from Table 8)
21	28	Iscan sign (+1/-1) (bit 1 from Table 8)

22	28	Jscan sign (+1/-1) (bit 2 from Table 8)
INFOGRID	Octet	GDS Content
if(sec2(4)==5) then	Polar stereographic grid	
1	7-8	I Dimension of the grid
2	9-10	J Dimension of the grid
3	11-13	Starting Latitude of the grid.
4	14-16	Starting Longitude of the grid.
5	17	Resolution and component flags.
6	18-20	Center longitude of the projection.
7	21-23	Grid-spacing in the I direction
8	24-26	Grid-spacing in the J direction
9	27	Projection center
10	28	Scanning mode (bit 3 from Table 8)
21	29	Iscan sign (+1/-1) (bit 1 from Table 8)
22	30	Jscan sign (+1/-1) (bit 2 from Table 8)
if(sec2(4)==50) then	Spherical Harmonic Coefficients	
INFOGRID	Octet	GDS Content
1	7-8	J-pentagonal resolution parameter
2	9-10	K-pentagonal resolution parameter
3	11-12	M-pentagonal resolution parameter
4	13	Spectral representation type (ON388 Table 9)
5	14	Coefficient storage mode (ON388 Table 10)
	15-32	Set to 0 (reserved)

## C.11 Additional information for setting up the READ\_GRIB routines for use on Linux Machines

A few steps were taken to modify the original READ\_GRIB routines to make them more compatible with Absoft, Lahey95, and other Linux (32-bit and 64-bit) based compilers. Here is a list of those steps:

Replaced the extensions of each \*.F file with \*.F90.

In the C-routine, cio.c, the following lines of code were added or modified:

- Line 19 Added “|| defined(ABSOFT)and || defined(LAHEY)”

In the Makefile, the following lines of code were added or modified:

- Line 19 Added “.F90” to .SUFFIXES rule
- Line 34-35 – Added “@echo make absoft” and “@echo make lahey”; resp.
- Line 71 (and following lines) Added flags and compiler names for ABSOFT:

**absoft:**

```

$(MAKE) $(LIBTARGET) \
"FC = f90" \
"FCFLAGS = -O -YEXT_NAMES=LCS -B108 -YCFRL=1 -YDEALLOC=ALL - \
DHIDE_SHR_MSG -DNO_SHR_VMATH -DABSOFT -DLITTLE_ENDIAN -DBIT -DBIT32" \
"CC = gcc" \
"CCFLAGS = -O -Wall -DABSOFT -DLITTLE_ENDIAN -DG_ENABLE_DEBUG=1" \
"CPP = /lib/cpp" \
"CPPFLAGS = -C -P -DBIT32"

lahey:
$(MAKE) $(LIBTARGETS) \
"FC = lf95" \
"FCFLAGS = -O -DBIT32 -DLINUX -DLAHEY -DLITTLE_ENDIAN" \
"CC = cc" \
"CCFLAGS = -O -DUSE_GCC -DLAHEY -DLITTLE_ENDIAN" \
"CPP = /lib/cpp" \
"CPPFLAGS = -C -P "

```

- Line 90 Changed “.F.o” to “.F90.o”
- Line 91 Removed “-d” from the rule

## C.12 Example of Fortran code that calls READ\_GRIB routines

```

!-- Initialize certain variables and parameters of GRIB file:
nunit = 10           ! Fortran unit number
ufn = nunit + 1     ! UNIX file descriptor associated with "nunit"
datarray = 0

!-- Open INPUT GRIB File:
call copen (nunit, ufn, trim(input_file)//char(0), 1, iret, 1)
print *, " ** Open File Code: ", iret

if ( iret > 0) then      ! Return File Error Code Number - IF FAILED TO OPEN!
    write(*,*) "STOPPING ROUTINE -- FILE NOT OPENED DUE TO CODE # :: ", iret
    stop
end if

!-- Read GRIB file:
call gribread ( ufn, datarray, ndata, ierr )

if ( ierr > 0) then    ! Return File Error Code Number - IF FAILED TO READ!
    write(*,*) "STOPPING ROUTINE -- FILE NOT READ DUE TO CODE # :: ", ierr

```

```
        stop
end if

!Print GRIB Header Information
do isec = 0, 2
    call gribprint (isec)
end do

do i = 1, ndata
    if (datarray(i) >0 ) then
        print *, i, datarray(i)
    end if
end do
```

Table 1: SEC0: GRIB Header Section 0 information

Number	Description
1	Length of a complete GRIB record
2	Grib Edition Number

Table 2: SEC1: GRIB Header Section 1 information

Octet	PDS content
1-3	Length of GRIB section 1 (3 bytes)
4	Parameter Table Version number
5	Center Identifier
6	Generating process Identifier
7	Grid ID number for pre-specified grids.
8	Binary bitmap flag:
9	Parameter ID Number and Units (ON388 Table 2)
10	Indicator of level type or layer (ON388 Table 3)
11	Level value (height or pressure), of the top value of a layer
12	Level value, but for bottom value of a layer ( 0 if NA ??)
13	Year (00-99)
14	Month (01-12)
15	Day of the month (01-31)
16	Hour (00-23)
17	Minute (00-59)
18	Forecast time unit: (ON388 Table 4)
19	Time period 1 (Number of Time Units Given in Octet 18)
20	Time period 2 or time interval between successive analyses
21	Time range indicator (ON833 Table 5)
22-23	Number included in average when Octet 21 (Table 5) indicates average or accumulation (otherwise set to 0)
24	Number missing from averages or accumulations
25	Century (Years 1999 and 2000 are century 20, 2001 is century 21)
26	Sub-center identifier
27-28	Decimal scale factor D. Negative value indicates setting high order bit in Octet 27 to 1 (“on”).
29	Is there a GDS (0=no, 1=yes; bit 1 of sec1(6)) Refer to Octet 8 above
30	Is there a BMS (0=no, 1=yes; bit 2 of sec1(6)) Refer to Octet 8 above

## References

- [1] GrADS. <http://grads.iges.org/grads/grads.html>.
- [2] Protex documenting system. <http://gmao.gsfc.nasa.gov/software/protex>.
- [3] CLM. <http://www.cgd.ucar.edu/tss/clm>.
- [4] DODS. <http://www.unidata.ucar.edu/packages/dods/>.
- [5] NOAH. <ftp://ftp.ncep.noaa.gov/pub/gcp/lidas/noahlsm/>.